(Tissue) P Systems with Decaying Objects

Rudolf Freund

Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
rudi@emcc.at

Abstract. Objects generated in P systems usually are assumed to survive as long as the computation goes on. In this paper, decaying objects are considered, i.e., objects only surviving a bounded number of computation steps. Variants of (tissue) P systems with decaying objects working in transition modes where the number of rules applied in each computation step is bounded, are shown to be very restricted in their generative power, i.e., if the results are collected in a specified output cell/membrane, then only finite sets of multisets can be generated, and if the results are specified by the objects sent out into the environment, we obtain the regular sets. Only if the decaying objects are regenerated within a certain period of computation steps, i.e., if we allow an unbounded number of rules to be applied, then computational completeness can be obtained, yet eventually more ingredients are needed for the rules than in the case of non-decaying objects, e.g., permitting and/or forbidden contexts. As special variants of P systems, catalytic P systems, P systems using cooperative rules, and spiking neural P systems are investigated.

1 Introduction

Cells in a living creature usually are not surviving the whole life time of this creature, e.g., the erythrocytes in the blood of humans have a life cycle of around four months. Hence, objects in systems modeling the functioning of structures of living cells – as for example, P systems – may be considered to have bounded time to survive, too. Formally, these objects will be called *decaying objects*, as their remaining time to survive without being involved in a rule is decreasing with each computation step.

In the area of P systems, decaying objects have already been considered in the case of spiking neural P systems: in [8] it was shown that with spiking neural P systems with decaying objects, which in this case are just the spikes stored in the neurons, only finite sets can be obtained, if the result is taken as the number of spikes contained in the output cell at the end of a computation; if the result is defined as the distance of the first two spikes sent to the environment from the output cell, then the linear sets of natural numbers can be characterized.

The idea of decaying objects also appears in the area of reaction systems, there being called the *assumption of no permanency*: an entity from a current configuration vanishes unless it is (re)produced by the application of a rule

E. Csuhaj-Varjú et al. (Eds.): CMC 2012, LNCS 7762, pp. 1–25, 2013.

(see [3]). In contrast to P systems, reaction systems work with sets of objects, i.e., multiplicities of objects are not taken into account, all ingredients are assumed to be available in a sufficient amount.

In the area of splicing systems, the effect of killing all strings not undergoing a splicing operation, turned out to be a powerful tool to control the evolution of splicing systems, in the end allowing for an optimal computational completeness result (see [14]) for time-varying distributed H systems using only one test tube (cell), to be compared with using splicing rules in P systems with only one membrane.

In this paper, the idea of decaying objects is extended to many other variants of (tissue) P systems. The combination of decaying objects and bounding the number of rules applicable in each computation step, drastically restricts the generative power of such systems usually known to be computationally complete with non-decaying objects: as in the case of spiking neural P systems, only finite sets can be generated if the output is taken as the number of (terminal) objects in an output cell/membrane, whereas a characterization of the regular sets is obtained if the output is defined as the collection or sequence of objects sent out to the environment. For catalytic P systems and for P systems using cooperative rules, computational completeness can be shown for several combinations of maximally parallel transition modes and halting conditions, yet in contrast to the computational completeness results for non-decaying objects, now for catalytic P systems permitting and forbidden context conditions are needed for the rules.

The rest of this paper is organized as follows: In the second section, we recall well-known definitions and notions. Then we define a general class of multiset rewriting systems containing, in particular, many variants of P systems and tissue P systems as well as even (extended) spiking neural P systems without delays, and formalize the idea of *decaying objects* in these systems. Moreover, we give formal definitions of the most important well-known *transition modes* (maximally parallel, minimally parallel, asynchronous, sequential) as well as the k-restricted minimally/maximally parallel transition modes and the parallel transition mode using the maximal number of objects; finally, we define variants of *halting*: the normal halting condition when no rules are applicable anymore (total halting), partial halting, adult halting, and halting with final states. In the third section, we first give some examples for P systems with decaying objects and discuss the restriction of the generative power of such systems in combination with transition modes only allowing for a bounded number of rules to be applied in parallel in each transition step in the general case. As a specific variant, first systems working in the sequential mode are considered; then, we investigate the effect of decaying objects in P systems working in the 1-restricted minimally parallel transition mode, i.e., spiking neural P systems without delays (in every neuron where a rule is applicable exactly one rule has to be applied) and purely catalytic P systems; finally, we investigate the k-restricted maximally parallel transition mode. In the fourth section, computational completeness results are established, especially for variants of catalytic P systems and of P systems using cooperative rules. An outlook

to future research topics for (tissue) P systems with decaying objects concludes the paper.

2 Definitions

In this section, we recall some well-known notions and define the basic model of networks of cells we use for describing different variants of (tissue) P Systems.

2.1 Preliminaries

The set of integers is denoted by $\mathbb{Z}$, the set of non-negative integers by $\mathbb{N}$. The interval $\{n \in \mathbb{N} \mid k \leq n \leq m\}$ is abbreviated by $[k..m]$. An *alphabet* V is a finite non-empty set of abstract *symbols*. Given V, the free monoid generated by V under the operation of concatenation is denoted by V^*; the elements of V^* are called strings, and the *empty string* is denoted by λ; $V^* \setminus \{\lambda\}$ is denoted by V^+. Let $\{a_1, \cdots, a_n\}$ be an arbitrary alphabet; the number of occurrences of a symbol a_i in a string x is denoted by $|x|_{a_i}$; the *Parikh vector* associated with x with respect to $a_1, \cdots, a_n$ is $\left(|x|_{a_1}, \cdots, |x|_{a_n}\right)$. The *Parikh image* of a language L over $\{a_1, \cdots, a_n\}$ is the set of all Parikh vectors of strings in L, and we denote it by $Ps(L)$. For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$.

A (finite) multiset over the (finite) alphabet V, $V = \{a_1, \cdots, a_n\}$, is a mapping $f : V \longrightarrow \mathbb{N}$ and represented by $\langle f(a_1), a_1\rangle \cdots \langle f(a_n), a_n\rangle$ or by any string x the Parikh vector of which with respect to $a_1, \cdots, a_n$ is $(f(a_1), \cdots, f(a_n))$. In the following we will not distinguish between a vector $(m_1, \cdots, m_n)$, its representation by a multiset $\langle m_1, a_1\rangle \cdots \langle m_n, a_n\rangle$ or its representation by a string x having the Parikh vector $\left(|x|_{a_1}, \cdots, |x|_{a_n}\right) = (m_1, \cdots, m_n)$. Fixing the sequence of symbols $a_1, \cdots, a_n$ in the alphabet V in advance, the representation of the multiset $\langle m_1, a_1\rangle \cdots \langle m_n, a_n\rangle$ by the string $a_1^{m_1} \cdots a_n^{m_n}$ is unique. The set of all finite multisets over an alphabet V is denoted by V°. For two multisets f_1 and f_2 from V° we write $f_1 \sqsubseteq f_2$ if and only if $f_1(a_i) \leq f_2(a_i)$ for all $1 \leq i \leq n$, and we say that f_1 is a submultiset of f_2.

A *context-free string grammar* is a construct $G = (N, T, P, S)$ where N is the alphabet of nonterminal symbols, T is the alphabet of terminal symbols, P is a set of context-free rules of the form $A \to w$ with $A \in N$, $w \in (N \cup T)^*$, and S is the start symbol. If all rules in P are of the forms $A \to bC$ with $A, C \in N$ and $b \in T^*$ or $A \to \lambda$ with $A \in N$, then G is called *regular*. A string v is derivable from a string u, $u, v \in (N \cup T)^*$, if $u = xAy$ and $v = xwy$ for some $x, y \in (N \cup T)^*$ and there exists a rule $A \to w$ in P; we write $u \Longrightarrow_G v$. The reflexive and transitive closure of the derivation relation $\Longrightarrow_G$ is denoted by $\Longrightarrow_G^*$. The string language generated by G is denoted by $L(G)$ and defined as the set of terminal strings derivable from the start symbol, i.e., $L(G) = \{w \mid w \in T^* \text{ and } S \Longrightarrow_G^* w\}$.

The family of regular, context-free, and recursively enumerable string languages is denoted by REG, CF, and RE, respectively. The family of finite languages is denoted by FIN, its complement, i.e., the family of co-finite languages,

by *co-FIN*. Two languages of strings or multisets L and L' are considered to be equal if and only if $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$.

For more details of formal language theory the reader is referred to the monographs and handbooks in this area such as [5] and [19]. Basic results in multiset rewriting can be found in [13]. Moreover, we assume the reader to be familiar with the main topics of membrane computing as described in the books [17] and [18]. For the actual state of the art in membrane computing, we refer the reader to the P Systems Webpage [20].

2.2 Register Machines

For our main results establishing computational completeness for specific variants of (tissue) P systems working in different transition modes, we will need to simulate register machines. A *register machine* is a tuple $M = (m, B, l_0, l_h, P)$, where m is the number of registers, P is the set of instructions bijectively labeled by elements of B, $l_0 \in B$ is the initial label, and $l_h \in B$ is the final label. The instructions of M can be of the following forms:

- $l_1 : (ADD\,(j)\,, l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$
  Increase the value of register j by one, and non-deterministically jump to instruction l_2 or l_3. This instruction is usually called *increment*.
- $l_1 : (SUB\,(j)\,, l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$
  If the value of register j is zero then jump to instruction l_3, otherwise decrease the value of register j by one and jump to instruction l_2. The two cases of this instruction are usually called *zero-test* and *decrement*, respectively.
- $l_h : HALT$. Stop 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 program counter, which indicates the next instruction to be executed. Computations start by executing the first instruction of P (labeled with l_0), and terminate with reaching a $HALT$-instruction. Without loss of generality, we assume the $HALT$-instruction to be the only instruction where the register machine halts.

Register machines provide a simple universal computational model [15]. In the generative case as we need it later, we start with empty registers, use the first two registers for the necessary computations and take as results the contents of the $m-2$ registers 3 to m in all possible halting computations; during a computation of M, only the registers 1 and 2 can be decremented, and when M halts in l_h, these two registers are empty. In the following, we shall call a specific model of P systems *computationally complete* if and only if for any such register machine M we can effectively construct an equivalent P system Π of that type simulating each step of M in a bounded number of steps and yielding the same results.

2.3 Networks of Cells

In [10], a formal framework for (tissue) P systems capturing the formal features of various transition modes was developed, based on a general model of

membrane systems as a collection of interacting cells containing multisets of objects, which can be compared with the models of networks of cells as discussed in [2] and networks of language processors as considered in [4]. Continuing the formal approach started in [10], k-restricted variants of the minimally and the maximally parallel transition modes were considered in [11], i.e., we considered a partitioning of the whole set of rules and allowed only multisets of rules to be applied in parallel which could not be extended by adding a rule from a partition from which no rule had already been taken into this multiset of rules, but only at most k rules could be taken from each partition. Most of the following definitions are taken from [7] and [11].

Definition 1. *A* network of cells with checking sets of degree $n \geq 1$ *is a construct* $\Pi = (n, V, T, w, R, i_0)$ *where*

1. n *is the number of cells;*
2. V *is a* finite alphabet*;*
3. $T \subseteq V$ *is the* terminal alphabet*;*
4. $w = (w_1, \ldots, w_n)$ *where* $w_i \in V^\circ$*, for each* i *with* $1 \leq i \leq n$*, is the* multiset initially associated to cell i*;*
5. R *is a finite set of* rules *of the form* $(E : X \to Y)$ *where* E *is a recursive condition for configurations of* Π *(see definition below), while* $X = (x_1, \ldots, x_n)$*,* $Y = (y_1, \ldots, y_n)$*, with* $x_i, y_i \in V^\circ$*,* $1 \leq i \leq n$*, are vectors of multisets over* V*; we will also use the notation*

$$(E : (x_1, 1) \ldots (x_n, n) \to (y_1, 1) \ldots (y_n, n))$$

   for a rule $(E : X \to Y)$*; moreover, the multisets* x_i *and* y_i *may be split into several parts or be omitted in case they equal the empty multiset;*
6. i_0 *is the* output cell.

A network of cells (in the following also simply called P *system*) consists of n cells, numbered from 1 to n and containing multisets of objects over V; initially cell i contains w_i. A *configuration* C of Π is an n-tuple $(u_1, \ldots, u_n)$ of multisets over V; the *initial configuration* of Π, C_0, is described by w, i.e., $C_0 = w = (w_1, \ldots, w_n)$. Cells can interact with each other by means of the rules in R. A rule $(E : (x_1, 1) \ldots (x_n, n) \to (y_1, 1) \ldots (y_n, n))$ is applicable to a configuration C with $C = (w_1, 1) \ldots (w_n, n)$ if and only if C fulfills condition E and $x_i \sqsubseteq w_i$, $1 \leq i \leq n$; its application means rewriting objects x_i from cells i into objects y_j in cells j, $1 \leq i, j \leq n$. In this paper, only regular conditions are considered, i.e., $E = (E_1, \ldots, E_n)$, where the E_i, $1 \leq i \leq n$, are regular sets; if $(w_1, 1) \ldots (w_n, n)$ describes the current configuration C, then C fulfills condition E if and only if $w_i \in E_i$, $1 \leq i \leq n$.

As specific conditions we will use *random contexts*, specified as sets of n-tuples of pairs $((P_1, Q_1), \ldots, (P_n, Q_n))$ where the P_i are the *permitting* and the Q_i are the *forbidden* contexts and are finite sets of multisets over V. An n-tuple $((P_1, Q_1), \ldots, (P_n, Q_n))$ allows for the application of the rule $(x_1, 1) \ldots (x_n, n) \to (y_1, 1) \ldots (y_n, n)$ to the configuration $(w_1, 1) \ldots (w_n, n)$ if, besides $x_i \sqsubseteq w_i$, $1 \leq i \leq n$, for all $u \in P_i$, $u \sqsubseteq w_i$ and for no $v \in Q_i$, $v \sqsubseteq w_i$.

The set of all multisets of rules *applicable* to C is denoted by $Appl\,(\Pi, C)$; a procedural algorithm how to obtain $Appl\,(\Pi, C)$ was described in [10].

For the specific *transition modes* to be defined in the following, the selection of multisets of rules applicable to a configuration C has to be a specific subset of $Appl\,(\Pi, C)$; for the transition mode ϑ, the selection of multisets of rules applicable to a configuration C is denoted by $Appl\,(\Pi, C, \vartheta)$.

Definition 2. *For the* asynchronous *transition mode (asyn),*

$$Appl\,(\Pi, C, asyn) = Appl\,(\Pi, C)\,,$$

i.e., there are no particular restrictions on the multisets of rules applicable to C.

Definition 3. *For the* sequential *transition mode (sequ),*

$$Appl\,(\Pi, C, sequ) = \{R' \mid R' \in Appl\,(\Pi, C) \;\; and \;\; |R'| = 1\}\,,$$

i.e., any multiset of rules $R' \in Appl\,(\Pi, C, sequ)$ has size 1.

The most important transition mode considered in the area of P systems is the *maximally parallel* transition mode where we only select multisets of rules R' that are not extensible, i.e., there is no other multiset of rules $R'' \supsetneqq R'$ applicable to C.

Definition 4. *For the* maximally parallel *transition mode (max),*

$$\begin{aligned} Appl\,(\Pi, C, max) = \{R' \mid\; & R' \in Appl\,(\Pi, C) \;\; and \; there \; is \\ & no \; R'' \in Appl\,(\Pi, C) \;\; with \; R'' \supsetneqq R'\}\,. \end{aligned}$$

For the *minimally parallel* transition mode, we need an additional feature for the set of rules R, i.e., we consider a partitioning Θ of R into disjoint subsets R_1 to R_p. Usually, this partition of R may coincide with a specific assignment of the rules to the cells. For any set of rules $R' \subseteq R$, let $\|R'\|$ denote the number of sets of rules R_j, $1 \leq j \leq p$, with $R_j \cap R' \neq \emptyset$.

In an informal way, the minimally parallel transition mode can be described as applying multisets such that from every set R_j, $1 \leq j \leq p$, at least one rule – if possible – has to be used. For the basic variant as defined in the following, in each transition step we choose a multiset of rules R' from $Appl\,(\Pi, C, asyn)$ that cannot be extended to $R'' \in Appl\,(\Pi, C, asyn)$ with $R'' \supsetneqq R'$ and such that $(R'' - R') \cap R_j \neq \emptyset$ and $R' \cap R_j = \emptyset$ for some j, $1 \leq j \leq p$, i.e., extended by a rule from a set of rules R_j from which no rule has been taken into R'.

Definition 5. *For the* minimally parallel *transition mode with partitioning Θ ($min(\Theta)$),*

$$\begin{aligned} Appl\,(\Pi, C, min(\Theta)) = \{R' \mid\; & R' \in Appl\,(\Pi, C, asyn) \;\; and \\ & there \; is \; no \; R'' \in Appl\,(\Pi, C, asyn) \\ & with \; R'' \supsetneqq R', \;\; (R'' \setminus R') \cap R_j \neq \emptyset \\ & and \; R' \cap R_j = \emptyset \; for \; some \; j, \;\; 1 \leq j \leq p\}\,. \end{aligned}$$

In the k-restricted minimally parallel transition mode, a multiset of rules from $Appl\,(\Pi, C, min(\Theta))$ can only be applied if it contains at most k rules from each partition R_j, $1 \leq j \leq p$.

Definition 6. *For the* k-restricted minimally parallel *transition mode with partitioning* Θ *(*$min_k(\Theta)$*),*

$$Appl\,(\Pi, C, min_k(\Theta)) = \{R' \mid R' \in Appl\,(\Pi, C, min(\Theta)) \text{ and } |R' \cap R_j| \leq k \text{ for all } j,\ 1 \leq j \leq p\}\,.$$

Each multiset of rules obtained by min_1 can be seen as a kind of basic maximally parallel vector; this interpretation also allows for capturing the understanding of the minimally parallel transition mode as introduced by Gheorghe Păun:

Definition 7. *For the* base vector minimally parallel *transition mode with partitioning* Θ *(*$min_{GP}(\Theta)$*),*

$$Appl\,(\Pi, C, min_{GP}(\Theta)) = \{R' \mid R' \in Appl\,(\Pi, C, min(\Theta)) \text{ and } R' \supseteq R'' \text{ for some } R'' \in Appl\,(\Pi, C, min_1(\Theta))\}\,.$$

In the k-restricted maximally parallel transition mode, a multiset of rules can only be applied if it is maximal but only contains at most k rules from each partition R_j, $1 \leq j \leq p$.

Definition 8. *For the* k-restricted maximally parallel *transition mode with partitioning* Θ *(*$max_k(\Theta)$*),*

$$Appl\,(\Pi, C, max_k(\Theta)) = \{R' \mid R' \in Appl\,(\Pi, C, max) \text{ and } |R' \cap R_j| \leq k \text{ for all } j,\ 1 \leq j \leq p\}\,.$$

Definition 9. *For the the* k*-restricted maximally parallel transition mode with only one partition,* $max_k(\{R\})$*, we also use the notion* k-restricted maximally parallel *transition mode (*max_k*), i.e., we get*

$$Appl\,(\Pi, C, max_k) = \{R' \mid R' \in Appl\,(\Pi, C, max) \text{ and } |R'| \leq k\}.$$

Example 1. Consider the P system

$$\Pi = (1, \{a, b\}, \{b\}, aa, \{a \to b\}, 1)\,.$$

Then the rule $a \to b$ (this notation represents the rule $(I : (a, 1) \to (b, 1))$ where I is the condition which is always fulfilled) must be applied twice in the maximally parallel transition mode, whereas in the minimally parallel mode it can be applied twice or only once. In the transition mode $min_1(\{R\})$, the rule is applied once, whereas in the mode max_1 no multiset of rules is applicable, because in the maximally parallel way the rule should be applied twice.

A variant of maximal parallelism requires the maximal number of objects to be affected by the application of a multiset of rules:

Definition 10. *For the transition mode requiring a* maximal number of objects *to be affected (maxobj),*

$$Appl\,(\Pi, C, maxobj) = \{R' \mid R' \in Appl\,(\Pi, C, asyn) \text{ and there is no } R'' \in Appl\,(\Pi, C, asyn) \text{ with } |Bound\,(R'')| > |Bound\,(R')|\},$$

where $Bound\,(R')$ *for any* $R' \in Appl\,(\Pi, C, asyn)$ *denotes the multiset of symbols from* C *affected by* R'*.*

For all the transition modes defined above, we now can define how to obtain a next configuration from a given one by applying an applicable multiset of rules according to the constraints of the underlying transition mode:

Definition 11. *Given a configuration* C *of* Π *and a transition mode* ϑ*, we may choose a multiset of rules* $R' \in Appl\,(\Pi, C, \vartheta)$ *in a non-deterministic way and apply it to* C*. The result of this* transition step *(or* computation step*) from the configuration* C *with applying* R' *is the configuration* $Apply\,(\Pi, C, R')$*, and we also write* $C \Longrightarrow_{(\Pi,\vartheta)} C'$*. The reflexive and transitive closure of the transition relation* $\Longrightarrow_{(\Pi,\vartheta)}$ *is denoted by* $\Longrightarrow^*_{(\Pi,\vartheta)}$*.*

Definition 12. *A* computation *in a P system* Π*,* $\Pi = (n, V, T, w, R, i_0)$*, starts with the initial configuration* $C_0 = w$ *and continues with transition steps according to the chosen transition mode* ϑ*; it is called* successful *if we reach a halting configuration* C *with respect to the halting condition* γ*.*

We now define several variants of halting conditions (e.g., see [7]): The usual way of considering a computation in a P system to be successful is to require that no rule can be applied anymore in the whole system, i.e., $Appl\,(\Pi, C, \vartheta) = \emptyset$ (we shall also use the notion *total halting* in the following, abbreviated by H). Taking a partitioning of the rule set (as for the minimally parallel mode), we may require that there exists an applicable multiset of rules containing one rule from each partition. A biological motivation for this variant of halting (*partial halting*, abbreviated by h) comes from the idea that a system may only survive as long as there are enough resources to give all subsystems the chance to continue their evolution. Computations also may be considered to be successful if at some moment a specific pattern appears (*halting with final states*, abbreviated by F). If the computation runs into an infinite loop with a specific configuration never changing again, we speak of *adult halting*, abbreviated by A.

$N(\Pi, \vartheta, \gamma, \rho)$ denotes the set of natural numbers computed by halting (with respect to the halting condition γ) computations of Π in the transition mode ϑ, with the numbers extracted from the output cell i_0 with respect to specific constraints specified by ρ, i.e., we either take the whole contents or only take the terminal symbols or else subtract a given constant l from the resulting numbers of objects in i_0. Moreover, we also consider an additional variant of obtaining the results by allowing the rules to send out multisets of objects into the environment, where we assume them to be collected as non-decaying objects; the

environment is considered as an additional cell labelled by 0, the rules therefore being of the form

$$(E : (x_1, 1) \dots (x_n, n) \to (y_0, 0)\,(y_1, 1) \dots (y_n, n)) .$$

We use the notation

$$NO_m C_n\,(\vartheta, \gamma, \rho)\,[\text{parameters for rules}]$$

to denote the family of sets of natural numbers generated by networks of cells $\Pi = (n, V, T, w, R, i_0)$ with $m = |V|$; γ specifies the way of halting, i.e., $\gamma \in \{H, h, A, F\}$; ϑ indicates one of the transition modes $asyn$, $sequ$, max, and max_k for $k \in \mathbb{N}$ as well as $min(p)$, $min_k(p)$, and $max_k(p)$ for $k \in \mathbb{N}$ with p denoting the number of partitions in the partitioning Θ; $\rho \in \{E, N, T\} \cup \{-l \mid l \in \mathbb{N}\}$ specifies how the results are taken from the number of objects in the specified output cell i_0 (if we take the whole contents, we use N; we take T if the results are taken modulo the terminal alphabet or else $-l$ when subtracting the constant l from the resulting numbers of objects in i_0) or else sent out to the environment (specified by E); the *parameters for rules* describe the specific features of the rules in R. If any of the parameters m and n is unbounded, we replace it by $*$. If we are not only interested in the total number of objects obtained in the output cell, but want to distinguish the different terminal symbols, in all the definitions given above we replace the prefix N by Ps indicating that then we get sets of vectors of natural numbers, corresponding with sets of Parikh vectors. In that case, the parameter $-l$ for ρ means that (at most) l nonterminal symbols may appear in the output cell, whereas N indicates that the number of additional nonterminal symbols is added to the first component of the result vector.

2.4 P Systems with Decaying Objects

In all the variants of P systems as defined above, we now may introduce the concept of decaying objects, i.e., for each object in the initial configuration and for each object generated by the application of a rule, we specify its decay d, i.e., the number of computation steps it may survive without having been affected by a rule. A decay of one means that the object b will die if it is not affected by a rule, which in some sense could be interpreted as the additional application of a rule $b \to \lambda$. We use the notation $b^{[k]}$ to specify that this object b may still survive k computation steps before having to be affected by the application of a rule. Assigning an additional value to each symbol b here is used to specify the remaining life time of this object in the system; another interpretation could be the concept of assigning a specific amount of energy; in this respect, there are similar approaches to be found in the literature, e.g., see the conformon P systems as introduced by Pierluigi Frisco ([18], chapter 10).

The contents of each membrane/cell of a P system has to be described by multisets of objects $b^{[k]}$, i.e., for each object b we also have to specify the remaining life time k. If $b^{[k]}$ occurring in a configuration C in cell j is not affected by a rule in the multiset of rules R' chosen from $Appl\,(\Pi, C, \vartheta)$, then this symbol appears

as $b^{[k-1]}$ in the next configuration C' derived from C by applying R', where formally we interpret $b^{[0]}$ as λ; in fact, applying R' in total can be interpreted as having applied a multiset of rules $R'^{[d]}$ obtained from R' by

a) interpreting each object b on the left-hand side as an object $b^{[k]}$ for some k with $1 \leq k \leq d$ and introducing each object c on the right-hand side as $c^{[d]}$;
b) adding a rule $\left(b^{[k]}, j\right) \rightarrow \left(b^{[k-1]}, j\right)$ for each object $b^{[k]}$ not affected by a rule from R' following the strategy in a).

In fact, in order to correctly specify these informal descriptions in the formal framework, we have to extend the definition of how the P system Π works with decaying objects of decay d as follows:

Definition 13. *For any (finite) alphabet V and any $d \in \mathbb{N}$,*

$$V^{\langle d \rangle} = \left\{ b^{[k]} \mid 1 \leq k \leq d \right\}.$$

The projection $h_d : \left(V^{\langle d \rangle}\right)^* \rightarrow V^*$ is defined by $h_d\left(b^{[k]}\right) = b$ for all $b \in V$ and $1 \leq k \leq d$. Given any additional finite set M, h_d can be extended to a projection $h_{d,M} : \left(V^{\langle d \rangle} \cup M\right)^* \rightarrow (V \cup M)^*$ by $h_{d,M}\left(b^{[k]}\right) = b$ for all $b \in V$ and $1 \leq k \leq d$ and $h_{d,M}(x) = x$ for all $x \in M$. If M is obvious from the context, we may write h_d instead of $h_{d,M}$ for short.

Given a P system Π and a decay d, we now are able to define the associated P system $\Pi^{[d]}$:

Definition 14. *For a P system $\Pi = (n, V, T, w, R, i_0)$ and a decay d, we define*

$$\Pi^{[d]} = \left(n, V^{\langle d \rangle}, T^{\langle d \rangle}, w^{[d]}, R^{[d]} \cup R_0^{[d]}, i_0\right)$$

and the rules in $R^{[d]}$ are obtained from the rules in R as follows:
For each rule

$$r = (E : (x_1, 1) \ldots (x_n, n) \rightarrow (y_0, 0)(y_1, 1) \ldots (y_n, n))$$

from R we take every rule

$$(x'_1, 1) \ldots (x'_n, n) \rightarrow (y_0, 0)\left(y_1^{[d]}, 1\right) \ldots \left(y_n^{[d]}, n\right)$$

with $h_d(x'_i) = x_i$, $1 \leq i \leq n$, into $R^{[d]}$, i.e., the right-hand sides are all equal, whereas the left-hand sides could be interpreted as the elements of

$$\left(h_{d,\{(,)\} \cup \{,\} \cup [1..n]}\right)^{-1} ((x_1, 1) \ldots (x_n, n)),$$

and we denote this set of rules obtained from r by $\hat{h}_d(r)$. Each newly generated object staying in the system gets the initial decay d; in the case objects are sent out into the environment, these are assumed to have no decay there, hence, we

just take the original multiset y_0 instead of $y_0^{[d]}$. A multiset of rules $\hat{R}'$ from $R^{[d]}$ is called an instance *of the rule set R' from R if and only if there exists a bijection $g : R' \to \hat{R}'$ such that $g(r) \in \hat{h}_d(r)$ for all $r \in R'$.*

Finally, we define

$$R_0^{[d]} = \left\{ \left(b^{[k]}, i \right) \to \left(b^{[k-1]}, i \right) \mid b \in V, 1 \leq k \leq d, 1 \leq i \leq n \right\},$$

i.e., the set of rules needed for reducing the remaining life time of objects not involved in a rule from $R^{[d]}$; $b^{[0]}$ formerly is to be interpreted as λ.

The P system Π and the associated P system $\Pi^{[d]}$ have to be considered in parallel to describe the computations in the P system Π with decaying objects of decay d:

Definition 15. *Given a P system $\Pi = (n, V, T, w, R, i_0)$ with decaying objects of decay d and a configuration C of $\Pi^{[d]}$ together with a transition mode ϑ, we may choose a multiset of rules $R' \in Appl(\Pi, h_d(C), \vartheta)$ in a non-deterministic way; then we have to find an instance $\hat{R}'$ of R' and a set $R'' \in R_0^{[d]}$ such that $\hat{R}' \cup R'' \in Appl\left(\Pi^{[d]}, C, maxobj\right)$ and apply $\hat{R}' \cup R''$ to C. The result of this* transition step *(or* computation step*) from the configuration C with applying $\hat{R}' \cup R''$ is the configuration $Apply^{[d]}(\Pi, C, R')$, and we also write $C \Longrightarrow_{\left(\Pi^{[d]}, \vartheta\right)} C'$. The reflexive and transitive closure of the transition relation $\Longrightarrow_{\left(\Pi^{[d]}, \vartheta\right)}$ is denoted by $\Longrightarrow^*_{\left(\Pi^{[d]}, \vartheta\right)}$.*

A computation *in the P system Π with decaying objects of decay d starts with the initial configuration represented by $w^{[d]}$ as a configuration in $\Pi^{[d]}$ and continues with transition steps according to the chosen transition mode ϑ as described above; it is called* successful *if we reach a configuration C such that $h_d(C)$ is a halting configuration of Π with respect to the halting condition γ; the results of this successful computation are taken from $h_d(C)$.*

Whereas the choice of the rule set to be applied only depends on the conditions given by the rules in R and the transition mode ϑ for Π (this justifies to not take into account the conditions E of rules $(E : X \to Y)$ from R in the corresponding rules of $R^{[d]}$), the total effect to the current configuration C represented as a configuration of $\Pi^{[d]}$ always affects all objects in C due to the mode *maxobj* used in $\Pi^{[d]}$. Although in the associated system $\Pi^{[d]}$ we always use the mode *maxobj*, no matter which transition mode is specified for Π itself, the results we obtain mostly will depend on the original transition mode specified for Π. Moreover, we emphasize that the condition of halting also only depends on the halting condition given for Π.

Remark 1. In order to make the condition for adult halting only depending on the halting condition given for Π, in this paper we assume a configuration C obtained by a computation in the P system Π with decaying objects of decay d to be a final one with respect to *adult halting* if and only if the set of multisets of rules applicable to $h_d(C)$ in Π is not empty, but the application of any of these

multisets of rules to $h_d(C)$ in Π yields $h_d(C)$ again. On the other hand, we might also assume a configuration C to be a final one with respect to adult halting if and only if there exists an infinite computation from C in the P system Π with decaying objects of decay d such that every configuration reachable along this computation is C; we might even require every possible computation starting from C to be infinite and never yielding another configuration than C. Although the arguments in the succeeding examples and proofs are given having in mind the first definition it is worth mentioning that the results hold true in each of the interpretations mentioned above.

It is easy to see that the use of decaying objects causes side-effects; for example, in the sequential mode *one* instance of a rule from R is applied, but in parallel *all* other remaining symbols are affected, too, by the decaying rules $\left(b^{[k]}, j\right) \rightarrow \left(b^{[k-1]}, j\right)$ applied in the associated system $\Pi^{[d]}$. The main problem with the application of these additional rules is that they allow symbols b to stay alive for a bounded period only without having been consumed by the application of another rule than these decaying rules. Another side-effect is the increase of non-determinism, as in the rules $(E : X \rightarrow Y)$ we specify the life time (decay) of the objects we generate in Y, but we do not specify which remaining life time the objects we take in X still should have; for example, the application of the rule $a \rightarrow b$ to the configuration $\left(a^{[2]}a^{[1]}, 1\right)$, in the sequential mode, yields the result $\left(b^{[2]}, 1\right)$ (assuming that a newly generated object starts with decay 2) if we consume the object $a^{[2]}$ by the application of the rule, whereas we obtain $\left(a^{[1]}b^{[2]}, 1\right)$ if we consume the object $a^{[1]}$ instead.

$N^{[d]}(\Pi, \vartheta, \gamma, \rho)$ $(Ps^{[d]}(\Pi, \vartheta, \gamma, \rho))$ denotes the set of (vectors of) natural numbers computed by halting (with respect to the halting condition γ) computations of Π with decaying objects of decay d in the transition mode ϑ, with the numbers extracted from the output cell i_0 with respect to the specific constraints specified by ρ. For the sets of (vectors of) natural numbers generated by P systems with decaying objects of decay $1 \leq k \leq d$ we now use the notation

$$YO_m^{[d]}C_n\,(\vartheta, \gamma, \rho)\,[\text{parameters for rules}]$$

with $Y \in \{N, Ps\}$, i.e., we add the superscript $[d]$ to specify the maximal life time of the objects.

3 P Systems with Decaying Objects and Transition Modes Bounding the Number of Rules in Applicable Multisets of Rules

In this section, we consider P systems having a constant K such that in each computation step the number of rules in an applicable multiset of rules is bounded by K.

3.1 Examples for P Systems with Decaying Objects

In this subsection, a few simple examples are exhibited to illustrate the effect of decays. For P systems with only one membrane/cell, we omit the indication of the cell number, i.e., instead of writing $(w, 1)$ we simply write w and instead of writing $(E : (x_1, 1) \to (y_1, 1))$ we may write $E : x_1 \to y_1$; moreover, if E is a condition which is always fulfilled, we may only write $x_1 \to y_1$.

Example 2. Consider the P systems

$$\Pi(d) = (1, \{s, a\}, \{a\}, s, \{s \to as, s \to \lambda\}, 1)$$

for $d > 1$. Then the only computations consist of applying n times the rule $s \to as$ and finally ending up with applying the rule $s \to \lambda$. For $n = 0$, we get $s^{[d]} \Longrightarrow_{(\Pi(d)^{[d]}, \vartheta)} \lambda$, for $1 \leq n \leq d$, we obtain the sequence of configurations

$$s^{[d]} \Longrightarrow^{n}_{(\Pi(d)^{[d]}, \vartheta)} a^{[d-n+1]} \dots a^{[d]} s^{[d]} \Longrightarrow_{(\Pi(d)^{[d]}, \vartheta)} a^{[d-n]} \dots a^{[d-1]},$$

whereas for $n > d$ we get

$$\begin{aligned} s^{[d]} &\Longrightarrow^{d+1}_{(\Pi(d)^{[d]}, \vartheta)} a^{[1]} a^{[2]} \dots a^{[d]} s^{[d]} \\ &\Longrightarrow^{*}_{(\Pi(d)^{[d]}, \vartheta)} a^{[1]} a^{[2]} \dots a^{[d]} s^{[d]} \Longrightarrow_{(\Pi(d)^{[d]}, \vartheta)} a^{[1]} \dots a^{[d-1]}. \end{aligned}$$

Hence, in sum we obtain

$$N^{[d]}(\Pi(d), \vartheta, \gamma, \rho) = \{n \mid 0 \leq n < d\},$$

for $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$ and any of the transition modes ϑ as defined in the preceding section as well as with γ denoting total halting, partial halting (the whole rule set forms the only partition), or halting with final states (defined by the regular set of multisets $\{a\}^{\circ}$, which in fact means the same as taking $\rho = T$). Therefore, the family of P systems $\Pi(d)$ with $d \in \mathbb{N}$ forms a very simple infinite hierarchy with respect to the decay d in any of these cases.

Example 3. Let M be a finite subset of T°. Consider the P system

$$\Pi(M) = (1, \{s\} \cup T, T, s, \{s \to w \mid w \in M\}, 1).$$

Obviously, $Ps^{[d]}(\Pi(M), \vartheta, \gamma, \rho) = M$ for $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$ and any of the transition modes ϑ as defined in the preceding section as well as with $\gamma \in \{H, h, F\}$; hence, for all $n, d \geq 1$,

$$PsO^{[d]}_{*}C_n(\vartheta, \gamma, \rho)\,[\text{ncoo}] \supseteq PsFIN,$$

where ncoo indicates (the use of) noncooperative rules (in general, a *noncooperative rule* is of the form $(I : (a, i) \to (y_1, 1) \dots (y_n, n))$ where a is a single symbol and I denotes the condition that is always fulfilled).

In the case of adult halting, we restrict ourselves to the transition modes $\vartheta \in \{max, maxobj\}$: If we add the rules $a \to a$ for all $a \in T$, then we obtain a P system $\Pi'(M)$ with $Ps^{[d]}(\Pi'(M), \vartheta, A, \rho) = M$ with respect to our convention to consider two multisets L and L' to be equal if and only if $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$. In that sense, we have

$$PsO_*^{[d]}C_n(\vartheta, A, \rho)\,[\text{ncoo}] \supseteq PsFIN.$$

Example 4. Let $G = (N, T, P, S)$ be a regular grammar (without loss of generality, we assume G to be reduced, i.e., every nonterminal symbol can be reached from the start symbol S and from every nonterminal symbol a terminal string can be derived). Consider the P system

$$\Pi(G) = (1, N \cup T, T, S, R, 1)$$

with

$$\begin{aligned} R = &\{(I : (A,1) \to (b,0)(C,1)) \mid A \to bC \in P\} \\ &\cup \{(I : (A,1) \to (\lambda,1)) \mid A \to \lambda \in P\}. \end{aligned}$$

Obviously, $Ps^{[d]}(\Pi(M), \vartheta, \gamma, E) = Ps(L(G))$ for any of the transition modes ϑ as defined in the preceding section as well as with $\gamma \in \{H, h, F\}$; hence, for all $n, d \geq 1$,

$$PsO_*^{[d]}C_n(\vartheta, \gamma, E)\,[\text{ncoo}] \supseteq PsREG.$$

In fact, the objects for the results of successful computations are collected in the environment, and all successful computations halt with empty cell 1.

Using the P system with decaying objects

$$\Pi'(G) = (1, N \cup T \cup \{F\}, T, S, R', 1)$$

with

$$\begin{aligned} R' = &\{(I : (A,1) \to (b,0)(C,1)) \mid A \to bC \in P\} \\ &\cup \{(I : (A,1) \to (F,1)) \mid A \to \lambda \in P\} \cup \{(I : (F,1) \to (F,1))\} \end{aligned}$$

we obtain $Ps^{[d]}(\Pi'(M), \vartheta, A, E) = Ps(L(G))$ for any of the transition modes ϑ as defined in the preceding section; all successful computations end up with looping in the final configuration F; hence, for all $n, d \geq 1$,

$$PsO_*^{[d]}C_n(\vartheta, A, E)\,[\text{ncoo}] \supseteq PsREG.$$

3.2 A General Lemma

The following result holds in general for all possible variants of rules as well as with all transition modes and halting conditions defined in the preceding section:

Lemma 1. *For all $d \geq 1$ and each $Y \in \{N, Ps\}$ as well as for ϑ being any transition mode guaranteeing that in each computation step only a bounded number of rules can be applied, we have that*

a) *for any halting condition* $\gamma \in \{H, h, F\}$ *and for any* $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$,

$$YO_*^{[d]}C_* (\vartheta, \gamma, \rho) [\textit{parameters for rules}] \subseteq YFIN$$

as well as,

b) *for any halting condition* $\gamma \in \{H, h, A, F\}$,

$$YO_*^{[d]}C_* (\vartheta, \gamma, E) [\textit{parameters for rules}] \subseteq YREG.$$

Proof (sketch). Let Π be an arbitrary P system with decaying objects of decays at most d, and let Z be the maximal number of objects generated by a rule from Π. Moreover, let K be the maximal number of rules applicable in a computation step in Π according to the transition mode ϑ. Then, no matter how many objects have been in the initial configuration, after d steps at most KdZ objects can be distributed over the cells of Π, as all the initial objects have either be used in the application of a rule or else have faded away due to their decay $\leq d$. Therefore, in any configuration computed in more than d steps, at most KdZ objects can be distributed over the cells of Π. No matter how these objects are distributed and how big is their actual decay, in sum only a finite number of different configurations may evolve from the initial configuration. Hence, also the number of results of successful computations in Π must be finite, which proves a).

For proving b), we construct a regular grammar $G = (N, T, P, S)$ as follows: All the different configurations that eventually may be computed from the initial configuration constitute the set of nonterminal symbols N; as shown before, their number is finite. The initial configuration is represented by the start symbol S. For each transition step from a configuration represented by the nonterminal A to a configuration represented by the nonterminal C thereby sending out the multiset w to the environment, we take the rule $A \to wC$ into P. If A represents a final configuration according to the halting condition γ, we take the rule $A \to \lambda$ into P. According to this construction it is easy to see that $Ps (L (G)) = Ps^{[d]} (\Pi, \vartheta, \gamma, E)$, which observation completes the proof. □

In combination with the Examples 3 and 4 we immediately infer the following characterizations of $YFIN$ and $YREG$, $Y \in \{N, Ps\}$:

Theorem 1. *For all* $d \geq 1$ *and each* $Y \in \{N, Ps\}$ *as well as for* ϑ *being any of the transition modes sequ,* max_k *for* $k \in \mathbb{N}$, $min_k(p)$, *or* $max_k(p)$ *for* $k \in \mathbb{N}$ *(with* p *denoting the number of partitions in the partitioning* Θ*),*

a) *for any halting condition* $\gamma \in \{H, h, F\}$ *and for any* $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$,

$$YO_*^{[d]}C_* (\vartheta, \gamma, \rho) [ncoo] = YFIN$$

as well as,

b) *for any halting condition* $\gamma \in \{H, h, A, F\}$,

$$YO_*^{[d]}C_* (\vartheta, \gamma, E) [ncoo] = YREG.$$

Proof (sketch). We only have to show that the given transition modes fulfill the condition needed for the application of Lemma 1. The maximal number K of rules applicable in Π according to the transition modes ϑ can be given as follows:

- for $\vartheta = sequ$, $K = 1$;
- for $\vartheta = max_k$, $k \in \mathbb{N}$, $K = k$;
- for $min_k(p)$ and $max_k(p)$, $k, p \in \mathbb{N}$, $K = kp$.

In all cases, the condition of Lemma 1 is fulfilled, which yields the inclusions $\subseteq$; the opposite inclusions follow by taking the P systems elaborated in Examples 3 and 4. □

In the remaining subsections of this section, we compare these results for specific variants of P systems with decaying objects from Theorem 1 with the computational completeness results obtained in [11] for the corresponding variants of P systems with non-decaying symbols.

3.3 Models for the 1-Restricted Minimally Parallel Transition Mode

In this subsection, as already described in [11], we use the ability of the 1-restricted minimally parallel transition mode to capture characteristic features of well-known models of P systems to compare the generative power of extended spiking neural P systems as well as of purely catalytic P systems with decaying and with non-decaying objects.

Extended Spiking Neural P Systems. We first consider extended spiking neural P systems (without delays), see [1], where the rules are applied in a sequential way in each neuron, but on the level of the whole system, the maximally parallel transition mode is applied – every neuron which may use a spiking rule has to spike, i.e., to apply a rule (see the original paper [12]). When partitioning the rule set according to the set of neurons, the application of the 1-restricted minimally parallel transition mode exactly models the original transition mode defined for spiking neural P systems.

An *extended spiking neural P system* (of degree $m \geq 1$) (in the following we shall simply speak of an *ESNP system*) is a construct $\Pi = (m, S, R, i_0)$ where

- m is the number of *neurons*; the neurons are uniquely identified by a number between 1 and m;
- S describes the *initial configuration* by assigning an initial value (of spikes) to each neuron;
- R is a finite set of *rules* of the form $\big(i, E/a^k \rightarrow P\big)$ such that $i \in [1..m]$ (specifying that this rule is assigned to neuron i), $E \subseteq REG\,(\{a\})$ is the *checking set* (the current number of spikes in the neuron has to be from E if this rule shall be executed), $k \in \mathbb{N}$ is the "number of spikes" (the energy) consumed by this rule, and P is a (possibly empty) set of *productions* of the

form (l, a^w) where $l \in [1..m]$ (thus specifying the target neuron), $w \in \mathbb{N}$ is the *weight* of the energy sent along the axon from neuron i to neuron l.
- i_0 is the *output neuron*.

A *configuration* of the ESNP system is described by specifying the actual number of spikes in every neuron. A *transition* from one configuration to another one is executed as follows: for each neuron i, we non-deterministically choose a rule $\left(i, E/a^k \to P\right)$ that can be applied, i.e., if the current value of spikes in neuron i is in E, neuron i "spikes", i.e., for every production (l, w) occurring in the set P we send w spikes along the axon from neuron i to neuron l. A *computation* is a sequence of configurations starting with the initial configuration given by S. An ESNP system can be used to generate sets from NRE (we do not distinguish between NRE and $RE\left(\{a\}\right)$) as follows: a computation is called *successful* if it halts, i.e., if for no neuron, a rule can be activated; we then consider the contents, i.e., the number of spikes, of the *output neuron* i_0 in halting computations.

We now consider the ESNP system $\Pi = (m, S, R, i_0)$ as a network of cells $\Pi' = (m, \{a\}, \{a\}, S, R', i_0)$ working in the 1-restricted minimally parallel transition mode, with

$$R' = \left\{ \left(E : \left(a^k, i\right) \to \left(a^{w_1}, l_1\right) \ldots \left(a^{w_n}, l_n\right)\right) \mid \left(i, E/a^k \to \left(l_1, a^{w_1}\right) \ldots \left(l_n, a^{w_n}\right)\right) \in R \right\}$$

and the partitioning R'_i, $1 \leq i \leq m$, of the rule set R' according to the set of neurons, i.e.,

$$R'_i = \left\{ \left(E : \left(a^k, i\right) \to \left(a^{w_1}, l_1\right) \ldots \left(a^{w_n}, l_n\right)\right) \mid \left(E : \left(a^k, i\right) \to \left(a^{w_1}, l_1\right) \ldots \left(a^{w_n}, l_n\right)\right) \in R' \right\}.$$

The 1-restricted minimally parallel transition mode chooses one rule – if possible – from every set R_i and then applies such a multiset of rules in parallel, which directly corresponds to applying one spiking rule in every neuron where a rule can be applied. Hence, it is easy to see that Π' and Π generate the same set from $RE\{a\}$ if in both systems we take the same cell/neuron for extracting the output. Due to the results valid for ESNP systems, see [1], we obtain:

Theorem 2. *For all $n \geq 3$,*

$$NRE = NO_1C_n\left(min_1\left(n\right), H, N\right)\left[ESNP\right].$$

In [8] the following results are shown for ESNP systems with decaying objects:

Theorem 3. *For all $n \geq 2$ and $d \geq 1$,*

a) $NFIN = NO_1^{[d]}C_n\left(min_1\left(n\right), H, N\right)\left[ESNP\right]$ *and*
b) $NREG = NO_1^{[d]}C_n\left(min_1\left(n\right), H, E\right)\left[ESNP\right]$.

Purely Catalytic P Systems. Already in the original papers by Gheorghe Păun (see [16] and also [6]), membrane systems with catalytic rules were defined, but computational completeness was only shown with using a priority relation on the rules. In [9] it was shown that only three catalysts are sufficient in one membrane, using only catalytic rules with the maximally parallel transition mode, in order to generate any recursively enumerable set of natural numbers. Hence, by showing that P systems with purely catalytic rules working in the maximally parallel transition mode can be considered as P systems working with the corresponding noncooperative rules in the 1-restricted minimally parallel transition mode when partitioning the rule sets for each membrane with respect to the catalysts, we obtain the astonishing result that in this case we get a characterization of the recursively enumerable sets of natural numbers by using only noncooperative rules.

A *noncooperative rule* is of the form $(I : (a, i) \to (y_1, 1) \dots (y_n, n))$ where a is a single symbol and I denotes the condition that is always fulfilled. A *catalytic rule* is of the form $(I : (c, i)\,(a, i) \to (c, i)\,(y_1, 1) \dots (y_n, n))$ where c is from a distinguished subset $C \subset V$ such that in all rules (noncooperative evolution rules, catalytic rules) of the whole system the y_i are from $(V \setminus C)^*$ and the symbols a are from $(V \setminus C)$. Imposing the restriction that the noncooperative rules and the catalytic rules in a network of cells allow for finding a hierarchical tree structure of membranes such that symbols either stay in their membrane region or are sent out to the surrounding membrane region or sent into an inner membrane, then we get the classical catalytic P systems without priorities. Allowing regular sets checking for the non-appearance of specific symbols instead of I, we even get the original P systems with priorities.

Catalytic P systems using only catalytic rules are called *purely catalytic P systems*. As we know from [9], only two (three) catalysts in one membrane are needed to obtain NRE with (purely) catalytic P systems without priorities working in the maximally parallel transition mode, i.e., we can write these results as follows (*cat* indicates that noncooperative and catalytic rules are allowed, whereas *pcat* indicates that only catalytic rules are allowed):

Theorem 4. *([9]) For all $n \in \mathbb{N}$ and $k \geq 2$, as well as $\gamma \in \{H, h, F\}$*

$$NRE = NO_*C_n\,(max, \gamma, -k)\,[cat_k] = NO_*C_n\,(max, \gamma, -(k+1))\,\left[pcat_{k+1}\right].$$

As the results can be collected in a second membrane without catalysts, we even have

$$NRE = NO_*C_{n+1}\,(max, \gamma, N)\,[cat_k] = NO_*C_{n+1}\,(max, \gamma, N)\,\left[pcat_{k+1}\right].$$

If we now partition the rule set in a purely catalytic P system according to the catalysts present in each membrane, this partitioning replaces the use of the catalysts when working in the 1-restricted minimally parallel transition mode, because by definition from each of these sets then – if possible – exactly one rule (as with the use of the corresponding catalyst) is chosen: from the set of purely catalytic rules R we obtain the corresponding set of noncooperative rules R' as

$$R' = \{(I : (a,i) \to (y_1,1)\dots(y_n,n)) \mid (I : (c,i)\,(a,i) \to (c,i)\,(y_1,1)\dots(y_n,n)) \in R\}$$

as well as the corresponding partitioning of R' as

$$R'_{i,c} = \{(I : (a,i) \to (y_1,1)\dots(y_n,n)) \mid (I : (c,i)\,(a,i) \to (c,i)\,(y_1,1)\dots(y_n,n)) \in R\}.$$

Considering purely catalytic P systems in one membrane, we immediately infer that when using the 1-restricted minimally parallel transition mode for a suitable partitioning of rules we only need noncooperative rules:

Corollary 1. *For all $n \in \mathbb{N}$ and $k \geq 3$ as well as $\gamma \in \{H, F\}$,*

$$NRE = NO_*C_n\,(min_1(k), \gamma, N)\,[ncoo].$$

On the other hand, when using the asynchronous, the sequential or even the maximally parallel transition mode, we only obtain regular sets (see [11]):

Theorem 5. *For each $Y \in \{N, Ps\}$, for any $\vartheta \in \{asyn, sequ, max\}$, any $\gamma \in \{H, h, A, F\}$, and any $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$,*

$$YREG = YO_*C_*\,(\vartheta, \gamma, \rho)\,[ncoo].$$

Combining the results of Theorem 5 with those from Theorem 1, we immediately obtain the following corollary for the sequential transition mode:

Corollary 2. *For any halting condition $\gamma \in \{H, h, A, F\}$, any $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$, and each $Y \in \{N, Ps\}$,*

$$YREG = YO_*^{[d]}C_*\,(sequ, \gamma, E)\,[ncoo] = YO_*C_*\,(sequ, \gamma, \rho)\,[ncoo],$$

for all $d \geq 1$.

For purely catalytic P systems with decaying objects, even in the maximally parallel transition modes *max* and *maxobj* the conditions of Lemma 1 are fulfilled, hence, we get the following results:

Theorem 6. *For all $n, d, k \geq 1$, each $Y \in \{N, Ps\}$, for any $\vartheta \in \{max, maxobj\}$, as well as for any halting condition $\gamma \in \{H, h, A, F\}$,*

$$YREG = YO_*^{[d]}C_n\,(\vartheta, \gamma, E)\,[pcat_k].$$

Theorem 7. *For all $n, d, k \geq 1$, each $Y \in \{N, Ps\}$, for any $\vartheta \in \{max, maxobj\}$, as well as for any halting condition $\gamma \in \{H, h, F\}$, and for any $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$,*

$$\begin{aligned} YFIN &= YO_*^{[d]}C_n\,(\vartheta, \gamma, -k)\,[pcat_k] = \cup_{k\geq 1} YO_*^{[d]}C_n\,(\vartheta, A, -k)\,[pcat_k] \\ &= YO_*^{[d]}C_{n+1}\,(\vartheta, \gamma, \rho)\,[pcat_k] = \cup_{k\geq 1} YO_*^{[d]}C_{n+1}\,(\vartheta, A, \rho)\,[pcat_k]. \end{aligned}$$

In all these systems with decaying objects, the catalysts are assumed to only have life time d, too.

3.4 The k-Restricted Maximally Parallel Transition Mode

In this subsection, we investigate the k-restricted maximally parallel transition mode. With this transition mode and cooperative rules, we again obtain computational completeness, a result which immediately follows from the results proved in the preceding section, i.e., from Theorem 4 and Corollary 1 (see [11]):

Corollary 3. *For all $n \geq 1$ and $k \geq 3$, as well as for any halting condition $\gamma \in \{H, h, F\}$,*

$$NRE = NO_*C_n\,(max_k, \gamma, -k)\,[coo] = NO_*C_n\,(max_k, \gamma, -k)\,[pcat_k]\,.$$

Yet in contrast to the results proved in the preceding section for the 1-restricted minimally transition mode, now with noncooperative rules we only obtain semilinear sets when using the k-restricted maximally parallel transition mode:

Theorem 8. *For all $n, k \geq 1$, each $Y \in \{N, Ps\}$, for every $k \in \mathbb{N}$ as well as any possible partitioning Θ of the rule sets in the P systems, i.e., for all $p \in \mathbb{N}$, for any halting condition $\gamma \in \{H, h, A, F\}$ and for any $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$,*

$$YREG = YO_*C_n\,(max_k(p), \gamma, \rho)\,[ncoo]\,.$$

Again, with decaying objects, the conditions of Lemma 1 are fulfilled, hence, we get the following results:

Theorem 9. *For all $n, d, k, p \geq 1$, each $Y \in \{N, Ps\}$, as well as for any halting condition $\gamma \in \{H, h, A, F\}$,*

$$YREG = YO_*^{[d]}C_n\,(max_k\,(p)\,, \gamma, E)\,[coo]\,.$$

Theorem 10. *For all $n, d, k, p \geq 1$, each $Y \in \{N, Ps\}$, as well as for any halting condition $\gamma \in \{H, h, F\}$, and for any $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$,*

$$YFIN = YO_*^{[d]}C_n\,(max_k\,(p)\,, \gamma, \rho)\,[coo] = \cup_{k \geq 1} YO_*^{[d]}C_n\,(max_k\,(p)\,, A, \rho)\,[coo]\,.$$

4 Computational Completeness Results for P Systems with Decaying Objects

In this section we prove computational completeness for catalytic P systems as well as for P systems using cooperative rules with decaying objects. Moreover, we only consider P systems with one membrane/cell.

Catalytic P systems can be seen as a specific variant of P systems using cooperative rules, hence, we first establish the computational completeness result for P systems using cooperative rules; when using arbitrary cooperative rules, additional ingredients such as context conditions can be avoided, yet only when using the transition mode *maxobj* instead of *max* as well as with adult halting or halting with final state:

Theorem 11. *For all $n \geq 1$ and all $d \geq 2$ as well as any $\gamma \in \{A, F\}$, any $\rho \in \{N, T\} \cup \{-l \mid l \in \mathbb{N}\}$, and each $Y \in \{N, Ps\}$,*

$$YRE = YO_*^{[d]}C_n\,(maxobj, \gamma, \rho)\,[coo]\,.$$

Proof (sketch). We only show $PsRE \subseteq PsO_*^{[2]}C_1\,(maxobj, \gamma, \rho)\,[\mathrm{coo}]$. The instructions of a register machine $M = (m, B, l_0, l_h, P)$ can be simulated by a P system $\Pi = (1, V, T, l_0, R, 1)$ with decaying objects of decay $d = 2$ using cooperative rules in the transition mode $maxobj$. As usual, the contents of a register j is represented by the corresponding number of copies of the symbol a_j; T consists of the symbols a_j, $3 \leq j \leq m$. For keeping the objects a_j, $1 \leq j \leq m$, alive, we use the rules $a_j \to a_j$.

- $l_1 : (ADD\,(j)\,, l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$,
  is simulated by the rules $l_1 \to l_2 a_j$ and $l_1 \to l_3 a_j$ in R.
- $l_1 : (SUB\,(j)\,, l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq 2$,
  is simulated in three steps:
  in the first step, the rule $l_1 \to l_1' h_j$ is used;
  in the second step, $l_1' \to \bar{l}_1$ is used, eventually in parallel with the rule $h_j a_j \to \bar{h}_j$ which is the crucial step of the simulation where we need the features of the transition mode $maxobj$ – it guarantees that for exactly one object a_j the rule $h_j a_j \to \bar{h}_j$ has priority over the rule $a_j \to a_j$ which involves less objects than the other one;
  finally, depending on the availability of an object a_j in the second step for the application of the rule $h_j a_j \to \bar{h}_j$, in the third step either $\bar{h}_j$ is present and the rule $\bar{l}_1 \bar{h}_j \to l_2$ is applied, or else h_j is still present so that the rule $\bar{l}_1 h_j \to l_3$ is used.
- $l_h : HALT$ is simulated by the rule $l_h \to \lambda$.

Collecting all objects used in the rules defined above, we get

$$V = B \cup \left\{l', \bar{l} \mid l \in B \setminus \{l_h\}\right\} \cup \left\{h_1, \bar{h}_1, h_2, \bar{h}_2\right\} \cup \{a_j \mid 1 \leq j \leq m\}\,.$$

At the end of a successful computation, only the objects a_j, $3 \leq j \leq m$, representing the result are present and kept in an infinite loop by the rules $a_j \to a_j$, hence, the condition for adult halting is fulfilled; in sum we have shown that $L\,(M) = Ps^{[2]}\,(\Pi, maxobj, A, \rho)$.

For halting with final states, we can use the condition that only the objects a_j, $3 \leq j \leq m$, may be present. It seems to be impossible to stop the application of the rules $a_j \to a_j$ without using context conditions (or priorities on the rules), hence, we have to restrict ourselves to the halting conditions A and F. □

The idea for simulating the SUB-instruction elaborated in the preceding proof does not work with the transition mode max as the application of the rule $h_j a_j \to \bar{h}_j$ cannot be enforced without giving it priority over the rule $a_j \to a_j$; on the other hand, when adding only these two priorities

$$h_j a_j \to \bar{h}_j > a_j \to a_j,\ 1 \le j \le 2,$$

(priorities were already used in the original paper [6]), then the rest of the proof of Theorem 11 also works with the transition mode *max*.

We now return to catalytic P systems and establish the computational completeness result for catalytic P systems with decaying objects using the standard transition mode *max* (and the standard total halting):

Theorem 12. *For all* $n \ge 1$, $k \ge 2$, *and all* $d \ge 2$ *as well as any* $\gamma \in \{H, h, A, F\}$, *any* $\rho \in \{T\} \cup \{-l \mid l \ge 0\}$, *and each* $Y \in \{N, Ps\}$,

$$YRE = YO_*^{[d]} C_n\,(max, \gamma, \rho)\,[cat_k]\,.$$

Proof (sketch). We only show $PsRE \subseteq PsO_*^{[2]} C_1\,(max, \gamma, -0)\,[\mathrm{cat}_2]$. The instructions of a register machine $M = (m, B, l_0, l_h, P)$ can be simulated by a P system $\Pi = (1, V, T, l_0 c_1 c_2, R, 1)$ with decaying objects of decay $d = 2$ using noncooperative and catalytic rules in the transition mode *max*. The contents of a register j is represented by the corresponding number of copies of the symbol a_j; T consists of the symbols a_j, $3 \le j \le m$. For keeping the objects a_j, $1 \le j \le m$, alive, we now use the rules with context conditions

$$\{(\{l'\}, \emptyset) \mid l \in B\} : a_j \to a_j.$$

- $l_1 : (ADD\,(j), l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \le j \le m$,
  is simulated in two steps by the rules
  $c_1 l_1 \to c_1 l_1'$ as well as $c_2 l_1' \to c_2 l_2 a_j$ and $c_2 l_1' \to c_2 l_3 a_j$ in R.
- $l_1 : (SUB\,(j), l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \le j \le 2$,
  is simulated in two steps, too:
  in the first step, the rule $c_1 l_1 \to c_1 l_1'$ and eventually the rule with context conditions

  $$\{(\{l_1\}, \emptyset) \mid l_1 : (SUB\,(j), l_2, l_3) \in R\} : c_2 a_j \to c_2 a_j'$$

  is used;
  in the second step, if a_j' is present, then the rules $c_1 a_j' \to c_1$ and $c_2 l_1' \to c_2 l_2$ are used in parallel; otherwise, only the rule with context conditions

  $$\left\{\left(\emptyset, \left\{a_j'\right\}\right)\right\} : c_2 l_1' \to c_2 l_3$$

  is used.
- $l_h : HALT$ is simulated by the sequence of rules $l_h \to l_h'$, $l_h' \to \lambda$.

Collecting all objects used in the rules defined above, we get

$$\begin{aligned} V = {} & B \cup \{l' \mid l \in B\} \cup \{c_1, c_2\} \\ & \cup \{a_j \mid 1 \le j \le m\} \cup \{a_1', a_2'\}\,. \end{aligned}$$

At the end of a successful computation, only the objects a_j, $3 \leq j \leq m$, representing the result are present and kept alive three steps when l_h appears, whereas the catalysts die after two steps and the computation successfully halts with no rule being applicable anymore; in sum we have shown that $L(M) = Ps^{[2]}(\Pi, max, H, -0)$. Partial halting with the trivial partitioning $\{R\}$ successfully stops as total halting. For halting with final states, we can use the condition that only the objects a_j, $3 \leq j \leq m$, may be present. Using again the rules $a_j \to a_j$ instead of the corresponding ones with context conditions, the condition for adult halting can be fulfilled. □

5 Summary and Future Research

The main purpose of this paper has been to investigate the effect of using decaying objects in contrast to the non-decaying objects used in most cases so far in the area of P systems. Many variants of P systems known to be computationally complete with non-decaying objects can be shown to only characterize the finite or the regular sets of multisets in combination with transition modes only allowing for the application of a bounded number of rules in each computation step. On the other hand, in combination with the maximally parallel mode, computational completeness can be obtained for catalytic and P systems using cooperative rules, respectively, yet only with also using permitting and forbidden contexts. As an interesting special result, computational completeness can be obtained for P systems using cooperative rules with the mode using the maximal number of objects, yet without needing context conditions.

With respect to the maximally parallel mode and the mode using the maximal number of objects, a lot of technical details remain for future research, especially concerning the need of using context conditions, not only in connection with catalytic P systems and P systems using cooperative rules, but also with many other variants of (static) P systems.

The effect of using decaying objects in combination with the asynchronous transition mode has been left open in this paper. With non-decaying objects, the asynchronous mode usually yields the same results as the sequential mode. Yet in connection with using decaying objects, the situation becomes more difficult, and although the generative power seems to become rather degenerate, precise characterizations might be challenging problems for future research.

In this paper, only generative P systems with decaying objects are investigated. Obviously, decaying objects can also be considered for accepting P systems as well as for P systems computing functions. In order to obtain high computational power, it again is necessary to keep objects alive for an arbitray long period of computation steps. Yet we may expect slightly different results compared with those obtained in the generative case, e.g., with transition modes only allowing for the application of a bounded number of rules in each computation step, specific variants of such P systems allow for at least accepting $FIN \cup co\text{-}FIN$.

The idea of decaying objects can be extended from static (tissue) P systems to dynamic P systems, where membranes (cells) may be newly generated and/or deleted. In addition, the idea of decaying entities can be extended to membranes, too, i.e., we may consider membranes (cells) only surviving for a certain number of computation steps. Moreover, in nature different types of cells have different life cycles; hence, it is quite natural to allow different objects to have different decays or even to allow to introduce different decays for the same object in different rules.

Another challenging problem is to find non-trivial infinite hierarchies with respect to the decay of objects for specific kinds of P systems with decaying objects; Example 2 shows a very simple example of such an infinite hierarchy with respect to the decay of the objects.

When going from multisets to sets of objects, another wide field of future research may be opened; in this case, reaction systems can be seen as very specific variants of such a kind of P systems.

Acknowledgements. The author gratefully acknowledges the useful suggestions and remarks from Erzsébet Csuhaj-Varjú, Marion Oswald, and Sergey Verlan during the preparation of this paper; special thanks go to Marion and Sergey, as many definitions and results presented in this paper came up from long discussions with them and were taken over from joint papers.

References

1. Alhazov, A., Freund, R., Oswald, M., Slavkovik, M.: Extended Spiking Neural P Systems. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2006. LNCS, vol. 4361, pp. 123–134. Springer, Heidelberg (2006)
2. Bernardini, F., Gheorghe, M., Margenstern, M., Verlan, S.: Networks of cells and Petri nets. In: Gutiérrez-Naranjo, M.A., Păun, G., Romero-Jiménez, A., Riscos-Núñez, A. (eds.) Proc. Fifth Brainstorming Week on Membrane Computing, Sevilla, pp. 33–62 (2007)
3. Brijder, R., Ehrenfeucht, A., Main, M.G., Rozenberg, G.: A tour of reaction systems. Int. J. Found. Comput. Sci. 22(7), 1499–1517 (2011)
4. Csuhaj-Varjú, E.: Networks of language processors, pp. 771–790 (2001)
5. Dassow, J., Păun, G.: Regulated Rewriting in Formal Language Theory. Springer (1989)
6. Dassow, J., Păun, G.: On the power of membrane computing. Journal of Universal Computer Science 5(2), 33–49 (1999)
7. Freund, R.: Transition and Halting Modes in (Tissue) P Systems. In: Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A., Rozenberg, G., Salomaa, A. (eds.) WMC 2009. LNCS, vol. 5957, pp. 18–29. Springer, Heidelberg (2010)
8. Freund, R., Ionescu, M., Oswald, M.: Extended spiking neural P systems with decaying spikes and/or total spiking. Int. J. Found. Comput. Sci. 19(5), 1223–1234 (2008)
9. Freund, R., Kari, L., Oswald, M., Sosík, P.: Computationally universal P systems without priorities: two catalysts are sufficient. Theoretical Computer Science 330, 251–266 (2005)

10. Freund, R., Verlan, S.: A Formal Framework for Static (Tissue) P Systems. In: Eleftherakis, G., Kefalas, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2007. LNCS, vol. 4860, pp. 271–284. Springer, Heidelberg (2007)
11. Freund, R., Verlan, S.: (Tissue) P systems working in the k-restricted minimally or maximally parallel transition mode. Natural Computing 10(2), 821–833 (2011)
12. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae 71(2-3), 279–308 (2006)
13. Kudlek, M., Martín-Vide, C., Păun, G.: Toward a Formal Macroset Theory. In: Calude, C.S., Pun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 123–134. Springer, Heidelberg (2001)
14. Margenstern, M., Rogozhin, Y., Verlan, S.: Time-varying Distributed H Systems with Parallel Computations: the Problem is Solved. In: Chen, J., Reif, J.H. (eds.) DNA9. LNCS, vol. 2943, pp. 48–53. Springer, Heidelberg (2004)
15. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice Hall, Englewood Cliffs (1967)
16. Păun, G.: Computing with membranes. J. of Computer and System Sciences 61(1), 108–143 (1998); and TUCS Research Report 208 (1998), `http://www.tucs.fi`
17. Păun, G.: Membrane Computing. An Introduction. Springer, Berlin (2002)
18. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
19. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages, vol. 3. Springer, Heidelberg (1997)
20. The P Systems Web Page, `http://ppage.psystems.eu`