First Steps Towards a Geometry of Computation

Michael Muskulus[1], Robert Brijder[2]

[1] Mathematical Institute
Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-mail: muskulus@math.leidenuniv.nl
[2] Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-mail: rbrijder@liacs.nl

Summary. We introduce a geometrical[3] setting which seems promising for the study of computation in multiset rewriting systems, but could also be applied to register machines and other models of computation. This approach will be applied here to membrane systems (also known as *P systems*) without dynamical membrane creation. We discuss the rôle of *maximum parallelism* and further simplify our model by considering only one membrane and sequential application of rules, thereby arriving at asynchronous multiset rewriting systems (*AMR systems*). Considering only one membrane is no restriction, as each static membrane system has an equivalent AMR system. It is further shown that AMR systems without a priority relation on the rules are equivalent to Petri Nets. For these systems we introduce the notion of *asymptotically exact computation*, which allows for stochastic appearance checking in a priori bounded (for some complexity measure) computations. The geometrical analogy in the lattice $\mathbb{N}_0^d$, $d \in \mathbb{N}$, is developed, in which a computation corresponds to a trajectory of a random walk on the directed graph induced by the possible rule applications. Eventually this leads to symbolic dynamics on the partition generated by shifted positive cones C_p^+, $p \in \mathbb{N}_0^d$, which are associated with the rewriting rules, and their intersections. Complexity measures are introduced and we consider non–halting, loop–free computations and the conditions imposed on the rewriting rules. Eventually, two models of information processing, *control by demand* and *control by availability* are discussed and we end with a discussion of possible future developments.

1 Introduction

There are many questions in biology that could benefit from a computer science theoretical treatment or a more mathematical approach. Computational biology [26] being a prominent example, there are also systems biology [33, 57] and artifical

[3] We should remark here that our use of the term "geometry of computation" has no connections with the more algebraically oriented work of Y. Lafont and F. Lamarche.

chemistry [58, 25] that are amenable to such an analysis. The growing field of *natural computing* [39, 8] studies biologically inspired models of computation [19].

In this work we propose a simple mathematical framework in which it might be possible to investigate some of these systems. It is based on *multiset rewriting* [9, 60, 42], as this captures essential features necessary for modelling molecular biology [27]: parallelism and non–determinism [2, 52].

Membrane systems (also known as *P systems*) were introduced by Gheorghe Păun [46] as a computational model of biochemical processing in cells, making use of a hierarchical structure of membranes and maximal parallel multiset rewriting (see below for formal definitions). They are a well researched model with numerous proposed variants (for the latest developments see [63]).

In our interdisciplinary effort to establish some link between computer science, biology, chemistry and mathematics, we will use a simplified version of membrane systems as our basic object of research. Standard membrane systems use the concept of *maximal parallelism*: The evolution of the system is governed by a global clock and in each timestep all the rewriting rules, that can be used, have to be used. The biological motivation for this is mass–action kinetics [61], but the strict form of *maximal parallelism* is unnatural biologically. Recently it has also been argued [3, 29] that mass–action kinetics is an unrealistic idealization itself (in molecular biology). *Maximal parallelism* is a source of computational power, as it allows for *appearance checking* [23]: testing whether there is a copy of an object inside a membrane.

Relaxing maximal parallelism is possible in a variety of ways. In *non–synchronized P systems* (Section 3.4.5 in [47]) some of the applicable rules can be used in parallel at each timestep, but are not forced to do so. In *sequential P systems* [28] exactly one rule is executed at each timestep, randomly chosen from the set of applicable rules. It has been shown [47] that non–synchronized P systems only generate number sets of context–free languages and are therefore not computationally complete. Completeness is achieved by the use of a priority relation on the rules, though. Sequential P systems can generate languages of partially blind counter automata [30] or of matrix grammars without appearance checking [28].

There are variants of these models, in which maximal parallelism is not abandoned, but rules are given execution times: *timed P systems* [18] assign a fixed integer duration to each rule, *clock–free P systems* [18] assign a random integer duration to each rule application (of another or of the same rule). Timed (respectively clock–free) P systems are called time–free (respectively clock–free) if their output (the language generated by the system [30]) is always the same, no matter how these durations are chosen. These notions correspond to the distinction between *physical parallelism* and *logical parallelism*, as pointed out by Banâtre and Métayer [9]: Physical parallelism refers to the underlying implementation of a computational process, whereas logical parallelism refers to the possibility of describing this process as several independent tasks. Clock–free systems and, to a lesser extent, also time–free systems have to rely only on logical parallelism and need to structure the computational process accordingly. This seems a natural

condition for modelling biological phenomena, as this implies *robustness* [10, 57] against random perturbations.

We propose a further simplification by studying *sequential* but non–deterministic systems that always produce the same output, calling these *asynchronous multiset rewriting systems* (*AMR systems*[4]). The loss in computational power can be partially compensated by the notion of *stochastic appearance checking*, in which a number of checks are made for the existence of an object. The probability of a correct computation can be regulated by increasing the number of checks, but a correct computation cannot be guaranteed anymore. Although this probability can be made arbitrarily close to unity, leading to the concept of *asymptotically exact computation*, this can only be done for bounded (in some complexity measure [12, 21]) computations. Still it seems to be a useful concept, achieving computational (quasi–) power with minimal effort, as is typical in biology. Furthermore, it could lead to intriguing biological considerations.

Considering the objective of making the computation amenable to analysis, there are at least two possibilities. One is to consider dependencies of rules among each other, leading to the notion of dependency graph [21]; the other is to consider the computational tree, tracing the system's configurations and all possible rule applications. So far, this has been mostly neglected in the literature on P systems (with the exception of [22]). The linearity of multisets, which has its analog in a similar linearity of rules, leads us to consider a linear space as the configuration space of the system. Introducing a numbering of the objects, this corresponds to a subset $\mathcal{S}$ of the regular lattice $\mathbb{N}_0^d$, where $d \in \mathbb{N}$ is the number of different objects. In this way, at present only P systems with static membrane structure can be modelled, but it might be possible to overcome this difficulty by considering the Banach space $\mathbb{N}^\infty$ of multisets over an infinite alphabet of objects and with an infinite number of potential rules. A different possibility would be to use a finite number of *types* of membranes [20].

The paper proceeds as follows: In Section 2 we define standard symbol–object P systems and AMR systems. In Section 3 we investigate the power of maximal parallelism and comment on how to circumvent it. The equivalence of static P systems with AMR systems, and the equivalence of the latter with Petri Nets is shown. Section 4 introduces the geometrical setting. Section 5 is concerned with appearance checking and complexity measures. Eventually, we arrive at a natural symbolic dynamics in Section 6. Section 7 contains some thoughts about the flow of control. Finally, we conclude with a discussion.

2 Membrane and Rewriting Systems

We begin by defining standard symbol–object P systems in a formal way. They are based on a hierarchical membrane structure (from [47]):

[4] The name *AMR system* has been preferred over the more economical seeming *MR system*, as this term already has an established meaning [54].

Definition 1 (Membrane structure). *A* membrane structure μ *is a rooted tree, where the nodes* $V(\mu)$ *are called* membranes*, the root* $V_0(\mu) = \mu_0$ *is called the* skin membrane*, and the leaves* $V_e(\mu)$ *are called* elementary membranes*.*

The contents of the membranes are characterized by multisets of objects.

Definition 2 (Multiset [60, 42]). *A* multiset *over a finite, non–empty set* O *is a mapping* $p : O \to \mathbb{N}_0$ *which assigns to each object* $o \in O$ *a positive multiplicity* $p(o) \in \mathbb{N}_0$*.*

Definition 3 (Generalized Multiset). *A* generalized multiset *over a finite, non–empty set* O *is a mapping* $p : O \to \mathbb{Z}$ *which assigns to each object* $o \in O$ *an integer "multiplicity"* $p(o) \in \mathbb{Z}$*.*

where we have used the following convention:

Convention 1 $\mathbb{N}_0 = \{0, 1, 2, \dots\}$.

We will sometimes write $\langle O, p\rangle$ for the multiset p over O, and $\mathbf{M}(O)$ for the set of all multisets over O.

Definition 4 (Operations on multisets). *Given two multisets* $\langle O, p\rangle$, $\langle O, q\rangle$*, we define the following operations:*

- *The sum* $\langle O, p\rangle + \langle O, q\rangle$ *is defined elementwise:*

$$(p + q)(a) := p(a) + q(a), \quad \forall a \in O$$

- *The* union $\langle O, p\rangle \sqcup \langle O, q\rangle$ *is defined as the elementwise maximum:*

$$(p \sqcup q)(a) := \max\left(p(a), q(a)\right), \quad \forall a \in O$$

- *The* intersection $\langle O, p\rangle \sqcap \langle O, q\rangle$ *is defined as the elementwise minimum:*

$$(p \sqcap q)(a) := \min\left(p(a), q(a)\right), \quad \forall a \in O$$

Definition 5 (Partial order on multisets). *We define a partial order* $<$ *on multisets such that*

$$\langle O, p\rangle < \langle O, q\rangle$$

if and only if

$$p(a) < q(a), \forall a \in O.$$

This extends in the obvious way to the order relations $\leq$, $>$, $\geq$*.*

Definition 6 (Numbering). *A* numbering *of a set* O *is an injective function* $c : O \to N$ *where* $N = \{1, 2, \dots, d\}$*,* $d = |O|$*.*

Using a numbering we can identify $\langle O, p\rangle$ with a vector p in some lattice $\mathbb{N}^d$, $d = |O|$. Generalized multisets will appear in the treatment of rules, and they correspond to elements of the integer lattice $\mathbb{Z}^d$.

The *configuration* of a standard symbol–object P system is characterized by the underlying cell:

Definition 7 (Cell). *A* cell *is a pair (μ, M), where μ is a membrane structure, and $M : V(\mu) \to \mathbf{M}(O)$ is a mapping assigning to each membrane the multiset of its contents in the set O.*

The basic computational step consists in the application of rules.

Definition 8 (Evolution rule). *An* evolution rule associated with a membrane *m is a triple $r = (d_r, v_r, \delta_r)$, where*

1. *d_r is a multiset over O of necessary* reactants.
2. *$v_r : V_c(\mu, m) \cup \{here, out\} \to \mathbf{M}(O)$ is a function assigning to each membrane reachable from m the products appearing in it after the application of the rule. $V_c(\mu, m)$ denotes the child membranes of m, and the special labels here and out are shortcuts for the actual membrane m and its parent in the membrane structure μ.*
3. *$\delta_r \in \{\delta, \neg\delta\}$ symbolizes membrane dissolution (δ) or keeping the membrane ($\neg\delta$).*

The availability of reactants is a necessary condition for the application of rules. The difference in existence or non–existence of an object allows for branching in the computational process. The evolution can be further regulated by a priority relation on the (collection of) rules:

Definition 9 (Collection of Rules). *A* collection of rules *is a function $R : V(\mu) \to 2^L$ which assigns to each membrane $m \in V(\mu)$ its associated rules, written as a set over the space of rules L.*

Remark 1 (Space of rules). The *space of rules* L is the space of rule triples, as given above. In the case of AMR systems (see below), the space of rules simplifies to some integer lattice $\mathbb{Z}^d$.

Remark 2 (Rule multiplicities). Considering collections of rules as multisets allows for *rule multiplicities*: Some rule can occur more than once. This viewpoint is beneficial for a stochastic treatment, where probabilities of rule applications depend on the rules' multiplicities. It is also the starting point for considerations of dynamical rule creation [5].

Definition 10 (Priority relation). *A* priority relation *over a collection of evolution rules R is a function σ with domain $V(\mu)$ such that for every membrane $m \in V(\mu)$, $\sigma(m)$ is a strict partial order over the multiset of rules $R(m)$ in this membrane.*

We are now ready to give the definition of a standard symbol–object P system. We will not use priorities in the following, therefore they have been excluded from the definition:

Definition 11 (Symbol–object P system). *A standard* symbol–object P system *is a 4–tuple $\Pi = (O, C_0, R, i_0)$, where:*

- *O is a non–empty, finite set of objects.*
- *$C_0 = (\mu_0, M_0)$ is the initial cell.*
- *R is a set of evolution rules associated with C_0.*
- *i_0 is a node of the rooted tree μ_0, called the* output membrane *of Π.*

Convention 2 (P system) *In the following we will use the abbreviation* P system *to refer to the standard symbol–object P system just defined.*

From now on, we are mostly interested in *static* P systems:

Definition 12 (Static P system). *A P system is called* static*, when the membrane structure does not change during the evolution.*

Is the membrane structure necessary? It is helpful for modelling issues, but as long as we have a *static* P system, we can always write these systems as an equivalent multiset rewriting system without any spatial structure, i.e., with *just one membrane*, by enlarging the object alphabet and the rules, as shown in Prop. 1 below.

With this in mind, we define in the following our main model of interest, the so–called *asynchronous multiset rewriting system* (AMR system). Using a numbering as before, we give the definition directly in the geometrical setting of some lattice $\mathbb{N}^d$.

Definition 13 (Dimension). *The* dimension *of an AMR system, denoted by d in the following, is the number of different objects used: $d = \mathrm{card}(O)$, where $O = \{1, 2, \ldots, d\}$.*

Retaining only rules without membrane dissolution, we arrive at:

Definition 14 (Simple rule). *A* simple rule *is a tuple (q_r, p_r), where*

1. *q_r is a multiset over O of* reactants*,*
2. *p_r is a multiset over O of* products*,*

and can be identified with a point $r = p_r - q_r$ in the integer lattice $\mathbb{Z}^d$, where d is the dimension of the system.

The *configuration* of an AMR system is completely characterized by the multiset $C : O \to \mathbb{N}$ of its object contents.

Definition 15 (Configuration). *The* configuration space *$\mathcal{S}$ of an AMR system is the space of multisets $\mathbf{M}(O)$ over the set of objects O and will be identified with the lattice of non–negative integers $\mathbb{N}_0^d$, where d is the dimension of the system.*

The application of a simple rule then simply amounts to vector addition: The configuration multiset C will be transformed via

$$C \Rightarrow C' = C + (p_r - q_r).$$

This is similar to a *vector addition system* [32, 53], the important difference being that in an AMR system the rules are conditional and can only be applied if the necessary reactants are available. The enhanced notion of a *vector addition system with states* [32] is equivalent to our model (as are Petri Nets, see below); but this will loose the geometrical intuition we want to develop — and which might lead to some new results.

As an enhancement to the model we can consider (generalized) inhibitors [14]:

Definition 16 (Enhanced rule). *An* enhanced rule *is a triple $r = (q, b, p)$, where*

1. *q is a multiset of necessary* reactants. *The rule is only* applicable *in a configuration C if $q \leq C$ (in the multiset sense).*
2. *b is a multiset of* inhibitors. *The rule is only applicable in a configuration C if $C(o) < b(o)$ for at least one object $o \in O$, i.e., if $q \not\leq C$.*
3. *p is a multiset of* products *which will be added to the configuration C after application of the rule.*

Such an inhibiting rule can also be written as

$$q \to p|_{\neg b},$$

and we will denote the left–hand–side (right–hand–side) of it by r_- (respectively r_+), i.e., $r_- = q$, $r_+ = p$.

The static description of our model can be given now; the dynamic scheme will be given in the next section:

Definition 17 (Asynchronous multiset rewriting system (AMR system), part 1). *An* AMR system *is a 3–tuple $\Pi = (d, C, R)$, where:*

- *$d \in \mathbb{N}$ is the dimension of the system.*
- *$C \in \mathcal{S}$ is the initial configuration of the system.*
- *R is a multiset of enhanced rules.*

Remark 3 (Priority relation). As an enhancement of the model, we could consider AMR systems with a priority relation on the rules, in analogy with P systems.

Now we can state:

Proposition 1 (Flattening the hierarchy). *For each static P system Π there exists an* equivalent *AMR system Π'.*

Proof. For each object $a \in O$ in Π define the objects a_i, $i \in \{1, \ldots, n\}$, in Π', where n is the number of membranes. Define the rules in Π' by substituting the appropriate reactants for the rules in Π: A rule $r = (d_r, v_r, \neg\delta)$ in a membrane m with reactant multiplicities $d_r(a) = j$ leads to the reactant multiplicities $q_r{}'(a_m) := j$ in Π'. The products will be mapped analogously: $v_r(k)(a) = j$ to $p_r{}'(a_k) := j$, and $v_r(\text{here})(a) = j$ will be mapped to $p_r{}'(a_m) := j$, and similar for the parent membrane. We have thus constructed the rule $r' = (q_r, \emptyset, p_r)$ in Π' equivalent to r in Π.

Proceeding in this way, we have established a one–to–one correspondence between the same symbol a in different membranes m_i, m_j and different symbols a_i, a_j in one membrane, exhibiting a kind of *duality* between membranes and rules. Configurations will be mapped in the obvious way, too. Concerning the inhibitors: standard P systems do not use them, so we do not need to map them. P systems with inhibitors [14] can be mapped analogously, though.

Given that each AMR system (without inhibitors) is a special P system, we have thus proved the equivalence of P systems and AMR systems. □

Remark 4 (Functoriality). The above equivalence can be made more precise in the context of category theory [38] as a functor from the category of P system configurations into the category of the corresponding configurations of the equivalent AMR system.

Remark 5 (Non–Uniqueness). The equivalent AMR system is only determined up to permutations of the representatives of objects in different membranes.

3 Parallelism and Synchronization

We now turn our attention to the application of the rules. This will be done in a non–deterministic, maximal parallel way, using a global clock with discrete timesteps. In membrane systems, this is defined as follows:

Definition 18 (Maximal parallel (P system case)). *At each timestep, all the objects in all the membranes evolve in a* maximal parallel *way, i.e., for each membrane m we have a multiset $\langle L, A_m \rangle$ over the space of rules L, such that it is* applicable *and* maximal*. A multiset of rules is* applicable*, if the union of the reactants of all the rules in it (counted with multiplicities) is contained in the membrane (as described by the cell). A multiset of rules is* maximal*, if there does not exist an applicable multiset of rules $\langle L, A'_m \rangle$ which is strictly larger (in the multiset sense).*

We have a similar notion for AMR systems:

Definition 19 (Maximal parallel (AMR system case)). *At each timestep, the AMR system's configuration C changes by addition of a vector $c \in \mathbb{Z}^d$ which fulfills the following condition: there exists a finite sequence of rules $s = (r_1, \ldots, r_n)$,*

$r_i = (q_i, b_i, p_i)$, *which is* applicable *and* maximal. *A sequence of rules is* applicable *if the sum of its reactants is contained in the multiset describing the system's configuration, and all of the rules are applicable (i.e., not inhibited). The sequence of rules is* maximal, *if there does not exist any larger applicable sequence* $(r_1, \ldots, r_n, \ldots, r_{n+k})$. *The change in the configuration will be described by* $c = \sum_i r_i = \sum_i (p_i - q_i)$.

This choice of dynamic scheme corresponds to the following algorithm:

i) Start by choosing one rule that is possible to apply randomly.
ii) Remove all the reactants and continue, until no more rules are applicable.
iii) Add all the products and move on to the next timestep.

The notions given above are equivalent under the above identification between static P systems and AMR systems:

Proposition 2 (Equivalence of Maximal Parallelism). *A maximal parallel application of rules in a static symbol object P system* Π *corresponds to a maximal parallel application of rules in the equivalent AMR system* Π'.

Proof. We have to establish an one–to–one correspondence between maximal parallel applications of rules, i.e., of a collection of applicable and maximal multisets $\{\langle L, A_m\rangle\}_{m\in V(\mu)}$ in Π to an equivalent maximal parallel application of rules in Π'. Define a sequence of rules $s = (r'_1, \ldots, r'_n)$ in the ARM system Π' such that for each rule r in some $(A_m)_{m\in V(\mu)}$ in Π with positive multiplicity $\sum_{m\in V(\mu)} A_m(r) = j$, $j > 0$, the corresponding rule r' in Π' occurs exactly j times in s. The order of rules in the sequence is insignificant.

(i) The sequence is applicable: Since the union of reactants is contained in the configuration $\mathcal{S}$ of Π, the equivalent configuration $\mathcal{S}'$ in Π' contains the equivalent objects. Concerning inhibitors: if none of the rules were inhibited in Π, they will also not be inhibited in Π'.

(ii) The sequence is maximal. If this were not the case, say we have a larger applicable sequence $s' = (r'_1, \ldots, r'_n, \ldots, r'_{n+k})$ in Π', the reactants of rule r'_{n+1} are contained in $\mathcal{S}'$. Thus the corresponding reactants of rule r_{n+1} in Π are contained in $\mathcal{S}$. The rule is also not inhibited in Π, and this leads to some larger, applicable multiset $\langle L, A'_m\rangle$, for some m, in the collection, which is a contradiction.

The other direction is proved analogously. □

As explained in the introduction, this dynamic scheme is biologically problematic. The following alternatives have therefore been proposed:

Definition 20 (Non–synchronized P system [47]). *A* non–synchronized *P system is a P system in which for each object* $a \in O$ *there exists the trivial rule* $a \to a$ *in each region.*

Now, because of non–deterministic choice, at each timestep a submultiset of all possible rules will be applied, i.e., the application of rules is not necessarily *maximal*

anymore. More precisely: for each rule application $r = \sum_i r_i$ in a non–synchronized P system there exists a corresponding maximal parallel rule application $r' = \sum_i r'_i$ such that for each r_i there exists a unique corresponding $r'_j = r_i$ for all i.

As proved in Section 3.4.5 of [47], the resulting model is only capable of generating the number sets of context–free languages [30] when priorities are not used. The use of priorities gives full computational power, though.

Another possibility to abandon maximum parallelism is to consider sequential systems:

Definition 21 (Sequential P system [28]). *A* sequential *P system is a P system in which at each timestep exactly one rule, chosen non–deterministically from all the applicable ones, is applied.*

Proposition 3 (Equivalence of Sequential and Non–synchronized AMR systems). *The notion of sequential and non–synchronized systems is the same for AMR systems without inhibitors (at the level of the output language).*

Proof. Consider a sequential application of rules $(r_1, \ldots, r_n)$. The same sequence can happen in a non–synchronized system in the possible case that only one rule is chosen at each timestep. Vice versa, if in one timestep we have an application of a set $\{r_1, \ldots, r_m\}$ of rules simultaneously in a non–synchronized system, this means that all the necessary reactants for the rules do exist. Sequential application of the same rules, *in any order*, is possible, since removing the reactants for any rule still leaves the necessary reactants for the other rules. It is only the availability of products that is delayed in comparison with the non–synchronized case.

We thus see that both systems generate the same language of possible outputs. □

Remark 6 (Parallelism at the level of rules). We can see from this proof, that rules corresponding to essentially parallel tasks can be applied in any order (cf. logical parallelism [9]). Essentially sequential tasks, i.e., tasks needing re–synchronization, can only be applied in a specific order. The prime mechanism for synchronization therefore is the production of objects that hitherto did not exist in the system and now allow a certain transition.

We now complete our definition of AMR systems by the appropriate dynamical scheme:

Definition 22 (Asynchronous multiset rewriting system (AMR system), part 2). *The computation in a* AMR system *proceeds by successive* sequential *state transitions in which in each configuration of the system one of the applicable enhanced rules is chosen in a* non–deterministic*, i.e., random, way.*

The AMR system is a special case of the more general *Abstract Rewriting System on Multisets* [59]. We think of the computation in an AMR system as a random walk on the directed graph induced by the structure of the rules.

Note that in such a system there is no notion of *external*, i.e., physical, time. "Time" can only be used *internally*, i.e., as a logical means to structure and synchronize the computation and refers only to the number of rule applications separating two configurations. This is similar to the notion of *time complexity* in the analysis of asynchronous distributed systems [6].

Finally, we show the fact that AMR systems without inhibitors and priorities are equivalent to Petri Nets [43]. The relationship between P systems and Petri Nets has already been commented on in [52], but in the simplified setting of AMR systems this equivalence is much more natural.

Proposition 4 (Equivalence of AMR systems and Petri Nets). *For each AMR system without inhibitors we can construct a naturally corresponding Petri Net and vice versa.*

Proof. Starting from an AMR system, for each of the objects we define a *place*, i.e., a node in the graph defining the structure of the Petri Net we are about to construct. For each of the rules, we construct a *transition*, i.e., a node in the graph, connected to places by directed edges such that the *in–places* (edges leading to this transition) correspond to the reactants and the *out–places* (edges leading away from this transition) correspond to the products. We assign *weights* to these edges which give the multiplicities of needed reactants or created products of the corresponding objects. A *marking* of this Petri Net, i.e., an assignment of non–negative integers to the places, corresponds to a configuration of the AMR system in the obvious way.

Analogously, we can construct an AMR system from a Petri Net. □

Corollary 1 (Power of AMR systems) *AMR systems without inhibitors are able to generate the context–sensitive languages as their* Szilard language *[55], i.e., the language generated by a labelling of the rules (see below), as this is shown for Petri Nets in [51].*

ARM systems with inhibitors, corresponding to extended Petri Nets*, have the full modelling power of the Turing machine [51].*

4 The Geometrical Setting

As seen above, configurations C of an AMR system correspond to points $C = (C_1, \ldots, C_d)$ in some lattice $\mathbb{N}^d$, with d the dimension of the system. Simple rules $r = (q, p)$, with $p = (p_1, \ldots, p_d) \in \mathbb{N}_0^d$, $q = (q_1, \ldots, q_d) \in \mathbb{N}_0^d$, correspond to points $q - p = (q_1 - p_1, \ldots, q_d - p_d)$ in the lattice $\mathbb{Z}^d$.

Definition 23 (Width). *The* width *of an AMR system, denoted by w in the following, is the number of rules present.*

We define a number of norms on the lattice $\mathbb{Z}^d$:

Definition 24 (p–Norm). *The* p–norm *of an element* $C = (C_1, \ldots, C_d) \in \mathcal{C}$ *is defined to be*

$$\|C\|_p = \left(\sum_{i=1}^{d} |C_i|^p \right)^{\frac{1}{p}}.$$

The familiar Euclidean norm $\|\cdot\|_2$ is a special case. Further examples for norms are *Manhattan distance* [48] and the *Maximum–Norm* $\|C\|_\infty = \max_i |C_i|$.

Definition 25 (Weight). *Define the* in–weight $|r_-|$ *and the* out–weight $|r_+|$ *of a rule* $r = (q, b, p)$ *to be*

$$|r_-| = \sum_{i=1}^{d} q_i, \quad |r_+| = \sum_{i=1}^{d} p_i.$$

These correspond to the 1–norm on $r_- = q$ *and* $r_+ = p$*, respectively.*

Define the weight *of a rule to be*

$$|r| = \max\left(|r_+|, |r_-|\right).$$

Definition 26 (Length). *Define the* length *of a rule* $r = (q, b, p)$ *to be* $\|p - q\|$*, where* $\|\cdot\|$ *is a norm on the lattice* $\mathbb{Z}^d$*.*

We now introduce the main geometric idea. It is based on concepts from convex analysis [13, 44].

Definition 27 (Cone). *Define*

$$C_q^+ := \{p \in \mathcal{C}, \mid p \geq q\},$$

where $p \geq q$ *iff* $p_i \geq q_i$ *for all* $i \in \{1, \ldots, d\}$*, to be a (positive, shifted)* cone *in configuration space.*

Definition 28 (Basic Cone of Applicability). *Define*

$$C_{(r)}^+ := C_{r_-}^+ = \{p \in \mathcal{C}, \mid p \geq r_-\},$$

to be the basic cone of applicability *of the simple rule* $r = (q, p)$*. For an enhanced rule* $r = (q, b, p)$ *we define similarly*

$$C_{(r)}^+ := C_q^+ \cap \left(C_b^+\right)^{\mathcal{C}},$$

where $\mathcal{C}$ *denotes the complement (see below). These cones, shifted to the origin, will be* convex *sets.*

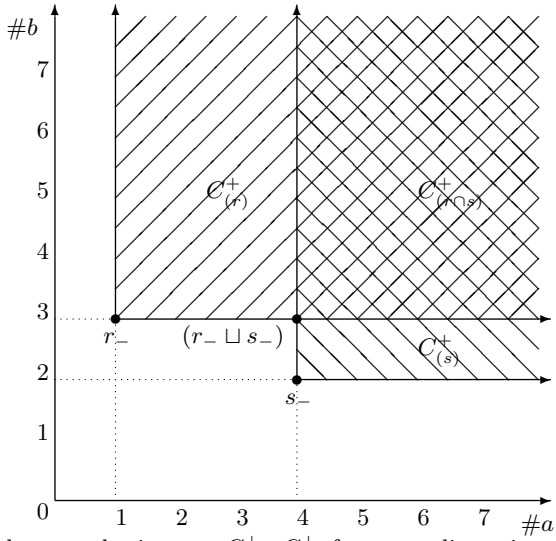

Fig. 1. Depicted are two basic cones $C^+_{(r)}, C^+_{(s)}$ for a two–dimensional system, corresponding to rules r, s with reactants $r_- = ab^3$, $s_- = a^4b^2$ and no inhibition. Their intersection $C^+_{(r\cap s)} := C^+_{(r)} \cap C^+_{(s)}$ is also shown.

Points in these cones correspond bijectively to system states in which a certain rule can be applied (notwithstanding some other rules that might also be applicable), i.e., they define bounds in configuration space, see Fig. 1.

The intersections and complements of these cones partition configuration space, leading to a natural symbolic dynamics, as shown below.

Remark 7 (Intersection). Intersection $C^+_{(r)} \cap C^+_{(s)}$ of two basic cones amounts to

$$C^+_{(r\,\cap\, s)} = C^+_{(r_- \,\sqcup\, s_-)} = \{p \in \mathcal{C} \,|\, p \geq (r_- \sqcup s_-)\},$$

in the case of simple rules.

The cone $C^+_{(r\,\cap\, s)}$ thus describes the part of configuration space in which both rules r and s are applicable.

Remark 8 (Complement). The *complement* of a cone $C^+_{(r)} \subseteq \mathcal{C}$ is defined as $(C^+_{(r)})^{\mathcal{C}} = \mathcal{S} - C^+_{(r)}$, and for a simple rule r this corresponds to bounding from above in *at least one dimension* (inhibition):

$$(C^+_{(r)})^{\mathcal{C}} = \{p \in \mathcal{S} \mid p_i < q_i, \text{ for some } i\}.$$

This is *not* the same as $C^-_{(r)} := \{p \in \mathcal{C} \mid p < r_-\}$.

Definition 29 (Cone of Applicability). *Define a* cone of applicability *to be any subset of configuration space that is attainable by finite intersection of basic cones* $\bigcap_{i\in S} C^+_{(r_i)}$, $S \subseteq \{1,\ldots,w\}$, *where* w *is the weight of the system.*

There are at most 2^w different cones of applicability, corresponding to all possible intersections.

Definition 30 (Generalized Cone). *Define a* generalized cone *to be any subset of configuration space that is attainable by a finite intersection of basic cones or their complements:*

$$S = \bigcap_{i\in A} C^+_{(r_i)} \bigcap_{j\in B} \left(C^+_{(r_j)}\right)^{\mathcal{C}},$$

where $A, B \subseteq \{1,\ldots,w\}$.

Considering AMR system computations, we would like to understand the behavior of AMR systems with input/output. Therefore we provide:

Definition 31 (Asynchronous multiset rewriting system (AMR system) with input/output). *Define two different objects to encode the input, respectively the output, of the system. The input objects must not be products of any rule of the system, and the output objects must not be reactants of any rule. This corresponds to the familiar construction of Turing machines with read–only input tape and write–only output tape used in defining* Logspace/Logtime *complexity classes [30]. The configuration space* $\mathcal{S}$ *can then be treated as a product:*

$$\mathcal{S} = \mathcal{S}_i \times \mathcal{S}_o \times \mathcal{S}_c,$$

where $\mathcal{S}_i$, $\mathcal{S}_o$ *are the one–dimensional input, resp. output, space, and* $\mathcal{S}_c$ *is the* $(d-2)$ *dimensional* internal configuration space. *The* internal dimension *of the system is* $d_c = d - 2$.

We can now characterize certain regions in configuration space:

Definition 32 (Halting state). *The* space of halting states *is defined as*

$$C^0 = \left(\bigcup_{i=1}^{w} C^+_{(r_i)}\right)^{\mathcal{C}},$$

where w *is the weight of the system corresponds to the subset of configuration space in which no rule is applicable at all.*

It is bounded in all dimensions that involve *non–cooperative rules*, i.e., rules with only one object $a \in O$ as necessary reactant. In general, it will be potentially unbounded.

Using just *communicative rules* [45], i.e., rules in which the total number of objects is conserved, the set of halting states will of course be bounded, as well as any computation.

5 Appearance Checking and Complexity Measures

5.1 Appearance Checking

Appearance checking refers to the task of determining if a certain object is present or not, and is the basis for control–flow branching. This is a non–trivial task in AMR systems. Checking for the existence of an object is easy if it exists, because we might have some rule that is then enabled. If none such object exists, though, there is no guaranteed way to arrive at a correct decision in finite time. We demonstrate this with the following example:

Example 1. Construct an AMR system that is able to detect whether an object a is given in the input or not. The input, or the initial state, will also contain a control object i_0. The output should consist of either an object y (for "yes") or an object n (for "no").

If inhibitors (or priorities) are used, appearance checking is trivial. The following rules check for the existence of object a, when the control object i_0 is present.

$$ai_0 \to y, \quad i_0 \to n|_{\neg a}$$

Without inhibitors, one possibility is using a stochastic cascade as follows:

$$\begin{aligned} i_0 &\to i_1 t_1, \\ ai_1 &\to y, \\ t_1 &\to t_2, \\ &\dots \\ t_{n-1} &\to t_n, \\ t_n i_1 &\to n. \end{aligned}$$

Assuming independent, equal probabilities for each rule, the probability for failure is $P(n|a) = (1/2)^{n+1}$ and can be made arbitrarily close to zero.

The probability of failure of the whole computation though depends (above all) on the number of times appearance checking is used. Clearly, if this number cannot be bounded, we cannot design a system such that it has a guaranteed minimum probability of success. We therefore turn to the issue of complexity measures.

5.2 Complexity Measures

The abstract theory of complexity [12] defines a complexity measure via the *Blum axioms*. We need to modify these in order to accommodate the non–deterministic setting.

Definition 33 (Complexity Measure). *A* complexity measure *for an AMR system is a partial function $\Theta : \mathcal{S} \to \mathbb{N}$ on the space of configurations $\mathcal{S}$ that satisfies the following two axioms:*

i) $\Theta(C)$ *is defined if* all *possible computations starting with the configuration* $C \in \mathcal{S}$ *are halting, and undefined if at least one of those computations is non–halting.*
ii) The predicate $\Theta(C) \leq y$ *is recursive, i.e., computable.*

The computability of bounds of complexity measures is achieved by the finiteness of the system states in a bounded computation.

In our geometric setting the most natural complexity measures are derived from the above given norms.

Definition 34 ($\|\cdot\|$–MAXSIZE**).**
Define $\|\cdot\|$*–MAXSIZE*(C) *to be the maximum of the norm* $\|\cdot\|$ *during the computation starting at* C*.*

Definition 35 (MAXTIME/MINTIME**).** *Define MAXTIME*(C)*/MINTIME*(C) *to be the maximal/minimal number of rule applications needed from configuration* C *until a halting configuration is reached.*

Remark 9 (Descriptional complexity). Apart from complexity measures for computations [50], there is also interest in *descriptional complexity*, i.e., complexity intrinsic to the system and its structure. Seeing computations basically as a way of compressing information, we arrive at the notion of Kolmogorov complexity [36] or the related Minimum Description Length principle [11]. We hope to return to these issues in forthcoming work.

Remark 10 (The number of objects needed). In a recent paper [48] it was noted that three different objects are enough to give a P system computational completeness In a way, this result seems to miss the point of descriptional complexity: The number of objects alone does not give enough information about the complexity of a standard P system, as the objects can have different effects in different membranes. Instead, one should either consider the *potential dimension* of the system, i.e., the number of different objects *times* the number of membranes, or investigate *uniform* P systems as defined below.

Definition 36 (Uniform P system). *An* uniform *P system is a P system in which the rules in each membrane are the same.*

Question 1. What is the computational power of uniform P systems and what is the minimum number of objects needed to achieve universality?

6 Symbolic Dynamics

The only necessity for a symbolic dynamics [37] is a (topological) partition of configuration space into disjoint sets.

Definition 37 (Natural partition). *We define the* natural partition $\mathcal{P}$ induced by the rules *as the partition defined by all possible intersections of the basic cones in the following way: Define*

$$C^+_{(i_1,\ldots,i_m)}, \quad 1 \leq i_j \leq k, \quad \forall j,$$

to be the cone

$$C^+_{(r_{i_1} \cap \cdots \cap r_{i_m})}$$

of applicability of the rules $r_{i_1}, \ldots, r_{i_m}$. There are at most 2^k distinct basic cones. Take $\mathcal{P}_0$ to be the set containing them. Some cones are included in other cones. Removing all overlaps by "clipping" the larger sets (in the inclusion ordering) we arrive at the desired partition $\mathcal{P}$. The set of halting states C^0 is included in this construction as the cone of applicability of no rule.

The symbolic dynamics on this partition is similar, but not quite the same, as the *Szilard language* [20, 55] of the system. The latter corresponds to a symbolic dynamics in which each rule is assigned a symbol and consecutive rule applications generate a string. The symbolic dynamics we propose has a more geometric origin and will generate a string in the following way: Each region of configuration space is assigned a symbol and each time a rule is applied the symbol corresponding to the region in which the system resides presently is output.

Maybe we can even combine these two approaches; together with the *rule dependency graph* [21], we have a lot of information to base our analysis on.

The basic problem with the computational tree is its size. In non–deterministic systems it can be infinite and still encode many useful properties of the system. We therefore need to find a way to reduce its complexity. The concept of *coverability tree* [43] from the theory of Petri nets [51] might hint at a method to achieve this.

Coming back to the dependency graph: we notice that in a cone $C^+_{(r \cap s)}$ with two possible rule applications, the rules can be either independent of each other, or be dependent of each other. The first case corresponds to the possibility of parallel application of the rules — in our sequential setting this means that the order of rule applications is unimportant: the rules *commute*. The second case is more complicated and we hope to comment on it in future work.

A simple condition on the rules is the following:

Definition 38 (Balanced rule). *A rule (q, b, p) is* balanced*, if at least one entry in its corresponding vector $p - q$ is negative.*

A rule that is not balanced will be called *exploding*, since it potentially leads to an unbounded, non–halting evolution of the system. A balanced rule on the other hand leaves every cone in a finite time, but interacting balanced rules can lead to zig–zag, re–entry and other higher order effects.

Checking the balance of all rules involves checking all possible non–negative, integer linear combinations of rules, leading to restricted linear systems of equations, a topic we would like to analyze in future work. Loop–free computations

have to fulfill some of these balance conditions in order to be halting. Strict conditions for the termination [24] of AMR systems might then be possible to derive. The theory of non–negative matrices [56] might be a useful tool in this analysis.

7 Control

In what way is information processed in our computational model? Following [34] we consider *control by demand* and *control by availability*.

Definition 39 (Control by Demand). *In the* control by demand *mode, control is mediated through the use of special control objects. If present, a certain computation will be initiated and eventually carried out (in parallel with all other computational tasks running). Either the system halts after this, or new control objects are introduced, which signal the end of the requested processing.*

This way of controlling a computation is familiar from most programming languages,, for example in the calling of subroutines. Its most striking feature is the distinction between data and control objects: Input data is sent into the system, but only after a control object is sent, will the processing start.

Definition 40 (Control by Availability). *In the* control by availability *mode there are no control objects. Data, i.e., objects created in the course of the computation, or given as initial input, is (potentially) processed as soon as it arrives in the system. In this way, the system "passes the data through" from one computational part to the next as in a FIFO queue (possibly splitting the data stream* $a \to bc$*, or synchronizing it* $ab \to c$*).*

This reminds of the functioning of Petri nets [43], of the way transcription of RNA takes place in parallel in the cell by multiple copies of RNA polymerase [4], or of models of neuronal processing [35]. In the latter, though, it can be argued that timing information is essential [1, 31].

Example 2 (Moving an object). We illustrate these concepts by the simple example of moving an input. Given an input a^n, $n \in \mathbb{N}$, the system should "compute" the output b^n.

- In *control by availability* mode, this is an easy task. The rule $a \to b$ takes care of it: As long as an object a is present, it will at some time be moved to an object b.
- In *control by demand* mode this is a non–trivial task, because the computation needs to "know" when it is finished in order to be able to pass the control on. We see the need for appearance checking again.

Summarizing, control by demand computations synchronize after completing subtasks with a kind of "handshake" that is passed on. Control by availability computations, on the other hand, do not synchronize explicitly. We therefore think that this model of "pass–through" computation captures the essence of biological computations better than the first one.

8 Discussion

We have introduced in this paper a geometric setting for the analysis of multiset rewriting systems. To simplify the analysis, we introduced asynchronous multiset rewriting systems (AMR systems) and showed how their configurations correspond to elements of a vector space and their evolution rules to vector addition.

This model is equivalent to vector addition systems (with states) and to Petri Nets, but the geometric view we have developed is new and interesting. Due to its simplicity, the model is amenable to analysis by geometrical, analytical and probabilistic methods; for example the theory of (Discrete) Convex Analysis [44] might find an application here. In this paper we have just given the basic definitions, but we hope to come back to most of these issues in forthcoming work.

The many connections with other areas of science are intriguing and we hope to gain in understanding from research in this area. Due to the closeness of our model with the theory of Petri nets [43] and Discrete Event Systems [7], we hope to arrive at some new insights in these areas; and eventually also into biological computation and control [57].

Acknowledgments

This research has been supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) under grant 635.100.006 in the Computational Life Sciences (CLS) programme. We would also like to thank the European Science Foundation (ESF) for inviting us to the Workshop EW04-134 on *Cellular Computing (Complexity Aspects)*, Sevilla, 2005, during which the main ideas of this paper have been developed. We thank all the participants for stimulating discussions, as well as G. Rozenberg, S.M. Verduyn–Lunel and H.J. Hoogeboom for their support.

References

1. M. Abeles: Time is precious. *Science*, 304 (2004), 523–524.
2. R. Adar, et. al: Stochastic computing with biomolecular automata. *Proc. Natl. Acad. Sci. USA*, 101, 27 (2004), 9960–9965.
3. P.S. Agutter, P.C. Malone, D.N. Wheatley: Diffusion theory in biology. A relic of mechanistic materialism. *Journal of the History of Biology*, 33 (2000), 71–111.
4. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter: *Molecular Biology of the Cell*. 4th edition, Garland Science, New York, 2002.
5. F. Arroyo, A. Baranda, J. Castellanos, Gh. Păun: Membrane computing: The power of (rule) creation. *J. Universal Computer Sci*, 8, 3 (2002), 369–381.
6. H. Attiya, J. Welch: *Distributed Computing. Fundamentals, Simulations and Advanced Topics*. Wiley, second ed., 2004.
7. F. Baccelli, G. Cohen, G.J. Olsen, J.-P. Quadrat: *Synchronization and Linearity. An Algebra for Discrete Event Systems*. Wiley Series in Probability and Mathematical Statistics, 1992.

8. D.H. Ballard: *An Introduction to Natural Computing*. MIT Press, 1997.
9. J.-P. Banâtre, D. Le Métayer: Programming by multiset transformation. *Comm. of the ACM*, 36, 1 (1993), 98–111.
10. N. Barkai, S. Leibler: Robustness in simple biochemical networks. *Nature*, 387 (1997), 913–917.
11. A. Barron, J. Rissanen, B. Yu: The minimum description length principle in coding and modeling. *IEEE Trans. Information Theory*, 44, 6 (1998), 2743–2760.
12. M. Blum: A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14, 2 (1967), 322–336.
13. J.M. Borwein, A.S. Lewis: *Convex Analysis and Nonlinear Optimization*. Vol. 3 of *CMS Books in Mathematics*, Springer-Verlag, Berlin, 2000.
14. P. Bottoni, C. Martín-Vide, Gh. Păun, G. Rozenberg: Membrane systems with promoters/inhibitors. *Acta Informatica*, 38, 10 (2002), 695–720 .
15. W. Brauer, W. Reising, G. Rozenberg, eds.: *Petri Nets. Applications and Relationships to Other Models of Concurrency*. LNCS 255, Springer-Verlag, Berlin, 1986.
16. C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Multiset Processing. Mathematical, Computer Science, Molecular Computing Points of View*, LNCS 2235, Springer-Verlag, Berlin, 2001.
17. M. Cavaliere, C. Martín-Vide, Gh. Păun, *Proceedings of the Brainstorming Week on Membrane Computing; Tarragona, February 2003*. Technical Report 26/03, Rovira i Virgili University, Tarragona, 2003.
18. M. Cavaliere, D. Sburlan: Time-independent P systems. In [40], 239–258.
19. J. Chen, D.H. Wood: Computation with biomolecules. *Proc. Natl. Acad. Sci. USA*, 97, 4 (2000), 1328–1330.
20. G. Ciobanu, Gh. Păun, Gh. Ştefănescu: Sevilla carpets associated with P systems. In [17], 135–140.
21. A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Weak metrics on configurations of a P system. In [49], 139–151.
22. A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Exploring computation trees associated with P systems. In [41], 196–204.
23. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.
24. J. Dershowitz, Z. Manna: Proving termination with multiset orderings. *Comm. of the ACM*, 22, 8 (1979), 465–476.
25. P. Dittrich, J. Ziegler, W. Banzhaf: Artificial chemistries – a review. *Artificial Life*, 7, 3 (2001), 225–275.
26. J. Doyle: Beyond the spherical cow. *Nature*, 411 (2001), 151–152.
27. M.J. Fischer, G. Malcolm, R.C. Paton: Spatio-logical processes in intracellular signalling. *BioSystems*, 55, 1-3 (2000), 83–92.
28. R. Freund: Asynchronous P systems and P systems working in the sequential mode. In [40], 36–62.
29. P.J. Halling: Do the laws of chemistry apply to living clls? *Trends in Biochemical Sciences*, 14, 8 (1989), 317–318.
30. J.E. Hopcroft, R. Motwani, J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2000.
31. J.J. Hopfield: Encoding for computations. Recognizing brief dynamical patterns by exploiting effects of weak rhythms on action-potential firing. *Proc. Natl. Acad. Sci. USA*, 101, 16 (2004), 5255–6260.

32. O.H. Ibarra, Z. Dang, O. Egecioglu: Catalytic membrane systems, semilinear sets, and vector addition systems. *Theoretical Computer Sci.*, 312, 2-3 (2004), 378–400.
33. H. Kitano: Computational systems biology. *Nature*, 420 (2002), 206–210.
34. W. Kluge: Reduction, data flow and control flow models of computation. In [15].
35. C. Koch: Computation and the single neuron. *Nature*, 385 (1997), 207–210.
36. M. Li, P. Vitányi: *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1993.
37. D. Lind, B. Marcus: *Symbolic Dynamics and Coding*. Cambridge Univ. Press., 1995.
38. S. MacLane: *Categories for the Working Mathematician*. Vol. 5 of GTM, Springer-Verlag, Berlin, second edition, 1998.
39. B.J. MacLennan: Natural computation and non-Turing models of computation. *Theoretical Computer Sci.*, 317, 1-3 (2004), 115–145.
40. G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, eds.: *Membrane Computing, International Workshop, WMC5, Milano, Italy, 2004, Selected Papers*. LNCS 3365, Springer-Verlag, Berlin, 2005.
41. G. Mauri, Gh. Păun, C. Zandron, eds.: *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*. Milano, 2004.
42. G.P. Monro: The concept of multiset. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 33, 2 (1987), 171–178.
43. T. Murata: Petri nets. Properties, analysis and applications. *Proc. Inst. Electr. Eng.*, 77, 4 (1989), 451–580.
44. K. Murota: *Discrete Convex Analysis*. Monographs on Discrete Mathematics and Applications, SIAM, 2003.
45. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–306.
46. Gh. Păun: Computing with membranes. *J. Computer and System Sciences*, 61, 1 (2000), 108–143.
47. Gh. Păun: *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
48. Gh. Păun, J. Pazos, M.J. Pérez-Jiménez, A. Rodríguez-Patón: Symport/antiport P systems with three objects are universal. *Fundamenta Informaticae*, 64, 1–4 (2005), 353–367.
49. Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds.: *Proceedings of the Second Brainstorming Week on Membrane Computing, Sevilla, February 2004*. Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004.
50. M.J. Pérez-Jiménez: An approach to computational complexity in membrane computing. In [40], 85–109.
51. J.L. Petersen. *Petri Nets Theory and the Modeling of Systems*. Prentice-Hall, 1981.
52. Z. Qi, J. You, H. Mao: P systems and Petri nets. In *Membrane Computing. Intern. Workshop, WMC2003, Tarragona*, LNCS 2933, Springer-Verlag, Berlin, 2004, 286–303.
53. C. Reutenauer: *The Mathematics of Petri Nets*. Prentice-Hall, 1990.
54. R. Rosen: A relational theory of biological systems. *Bull. Math. Biophysics*, 20 (1958), 245–260.
55. A. Salomaa: *Formal Languages*. Academic Press, 1973.
56. E. Seneta: *Non-negative Matrices and Markov Chains*. Springer Series in Statistics, second edition, 1981.
57. E.D. Sontag: Some new directions in control theory inspired by systems biology. *IEE Journal of Systems Biology*, 1, 1 (2004), 9–18.

58. P. Speroni di Fenizio, P. Dittrich: Artificial chemistry's global dynamic. Movements in the lattice of organization. *Journal of Three Dimensional Images*, 16, 4 (2002), 160–163.
59. Y. Suzuki, Y. Fujiwara, J. Takabayashi, H. Tanaka: Artificial life applications of a class of P systems: Abstract rewriting systems on multisets. In [16], 299–346.
60. A. Syropoulos: Mathematics of multisets. In [16], 347–358.
61. O. Temkin, A. Zeigarnik, D. Bonchev: *Chemical Reaction Networks. A Graph-Theoretical Approach*. CRC Press, 1996.
62. M. Walicki, S. Medal: Algebraic approaches to nondeterminism. An overview. *ACM Computing Surveys*, 29, 1 (1997), 30–81.
63. The P Systems Web Page: `http://psystems.disco.unimib.it`.