Unfair P Systems

Artiom Alhazov[1,2⋆], Rudolf Freund[3], and Sergiu Ivanov[4,5]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău, MD-2028, Moldova
artiom@math.md

[2] Key Laboratory of Image Information Processing
and Intelligent Control of Education Ministry of China,
School of Automation
Huazhong University of Science and Technology
Wuhan 430074, China

[3] Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Vienna, Austria
rudi@emcc.at

[4] LACL, Université Paris Est – Créteil Val de Marne
61, av. Général de Gaulle, 94010, Créteil, France
sergiu.ivanov@u-pec.fr

[5] TIMC-IMAG/DyCTiM, Faculty of Medicine of Grenoble
5 avenue du Grand Sablon, 38700, La Tronche, France
sergiu.ivanov@univ-grenoble-alpes.fr

Summary. We introduce a novel kind of P systems in which the application of rules in each step is controlled by a function on the applicable multisets of rules. Some examples are given to exhibit the power of this general concept. Moreover, for three well-known models of P systems we show how they can be simulated by P systems with a suitable fairness function.

1 Introduction

Membrane computing is a research field originally founded by Gheorghe Păun in 1998, see [5]. Membrane systems (also known as P systems) are a model of computing based on the abstract notion of a membrane and the rules associated to it which control the evolution of the objects inside. In many variants of P systems,

⋆ The work is supported by National Natural Science Foundation of China (61320106005 and 61033003) and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012).

the objects are plain symbols from a finite alphabet, but P systems operating on more complex objects (e.g., strings, arrays) have been considered, too, e.g., see [2].

A comprehensive overview of different flavors of membrane systems and their expressive power is given in the handbook, see [6]. For a state of the art snapshot of the domain, we refer the reader to the P systems website [9] as well as to the bulletin series of the International Membrane Computing Society [8].

In this paper we introduce a novel kind of P systems in which the application of rules in each step is controlled by a function on the applicable multisets of rules, possibly also depending on the current configuration; we call this function the *fairness function*. In the standard variant, the fairness function will be used to choose those applicable multisets for which the fairness function yields the minimal value.

After recalling some preliminary notions and definitions in the next section, in Section 3 we will define our new model of *fair P systems* and give some examples to exhibit the power of this general concept. In Section 4, for three well-known models of P systems we will show how they can be simulated by P systems with a suitable fairness function. Future research topics finally are touched in Section 5.

2 Preliminaries

In this paper, the set of positive natural numbers $\{1, 2, \dots\}$ is denoted by $\mathbb{N}_+$, the set of natural numbers also containing 0, i.e., $\{0, 1, 2, \dots\}$, is denoted by $\mathbb{N}$. The set of integers denoted by $\mathbb{Z}$.

An *alphabet* V is a finite set. A (non-empty) *string* s over an alphabet V is defined as a finite ordered sequence of elements of V.

A *multiset* over V is any function $w : V \to \mathbb{N}$; $w(a)$ is the *multiplicity* of a in w. A multiset w is often represented by one of the strings containing exactly $w(a)$ copies of each symbol $a \in V$; the set of all these strings representing the multiset w will be denoted by $str(w)$. The set of all multisets over the alphabet V is denoted by V°. By abusing string notation, the empty multiset is denoted by λ.

The families of sets of Parikh vectors as well as of sets of natural numbers (multiset languages over one-symbol alphabets) obtained from a language family F are denoted by PsF and NF, respectively. The family of recursively enumerable string languages is denoted by RE.

For further introduction to the theory of formal languages and computability, we refer the reader to [6, 7].

2.1 (Hierarchical) P Systems

A *hierarchical P system* (P system, for short) is a construct

$$\Pi = (O, T, \mu, w_1, \dots, w_n, R_1, \dots R_n, h_i, h_o),$$

where O is the alphabet of objects, $T \subseteq O$ is the alphabet of terminal objects, μ is the membrane structure injectively labeled by the numbers from $\{1, \dots, n\}$

and usually given by a sequence of correctly nested brackets, w_i are the multisets giving the initial contents of each membrane i ($1 \leq i \leq n$), R_i is the finite set of rules associated with membrane i ($1 \leq i \leq n$), and h_i and h_o are the labels of the input and the output membranes, respectively ($1 \leq h_i \leq n$, $1 \leq h_o \leq n$).

In the present work, we will mostly consider the *generative case*, in which Π will be used as a multiset language-generating device. We therefore will systematically omit specifying the input membrane h_i.

Quite often the rules associated with membranes are multiset rewriting rules (or special cases of such rules). Multiset rewriting rules have the form $u \to v$, with $u \in O^o \setminus \{\lambda\}$ and $v \in O^o$. If $|u| = 1$, the rule $u \to v$ is called *non-cooperative*; otherwise it is called *cooperative*. Rules may additionally be allowed to send symbols to the neighboring membranes. In this case, for rules in R_i, $v \in O \times Tar_i$, where Tar_i contains the targets *out* (corresponding to sending the symbol to the parent membrane), *here* (indicating that the symbol should be kept in membrane i), and in_j (indicating that the symbol should be sent into the child membrane j of membrane i).

In P systems, rules are often applied in the maximally parallel way: in any derivation step, a non-extendable multiset of rules has to be applied. The rules are not allowed to consume the same instance of a symbol twice, which creates competition for objects and may lead to the P system choosing non-deterministically between the maximal collections of rules applicable in one step.

A computation of a P system is traditionally considered to be a sequence of configurations it successively can pass through, stopping at the halting configuration. A halting configuration is a configuration in which no rule can be applied any more, in any membrane. The result of a computation of a P system Π as defined above is the contents of the output membrane h_o projected over the terminal alphabet T.

Example 1. For readability, we will often prefer a graphical representation of P systems; moreover, we will use labels to identify the rules. For example, the P system $\Pi_1 = (\{a, b\}, \{b\}, [\,_1\,]_1, a, R_1, 1)$ with the rule set $R_1 = \{1 : a \to aa, 2 : a \to b\}$ may be depicted as in Figure 1.

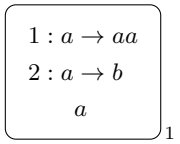

Fig. 1. The example P system Π_1

Due to maximal parallelism, at every step Π_1 may double some of the symbols a, while rewriting some other instances into b.

Note that, even though Π_1 might express the intention of generating the set of numbers of the powers of two, it will actually generate the whole of $\mathbb{N}_+$ (due to

the halting condition). Indeed, for any $n \in \mathbb{N}_+$, a^n can be generated in n steps by choosing to apply, in the first $n-1$ steps, $1 : a \to aa$ to exactly one instance of a and $a \to b$ to all the other instances, and by applying $2 : a \to b$ to every a in the last step (in fact, for $n > 1$, in each step except the last one, in which $2 : a \to b$ is applied twice, both rules are applied exactly once, as exactly two symbols a are present, whereas all other symbols are copies of b). □

While maximal parallelism and halting by inapplicability have been standard ingredients from the beginning, various other derivation modes and halting conditions have been considered for P systems, e.g., see [6].

2.2 Flattening

The folklore flattening construction (see [6] for several examples as well as [3] for a general construction) is quite often directly applicable to many variants of P systems. Hence, also for the systems considered in this paper we will not explicitly mention how results are obtained by flattening.

3 P Systems with a Fairness Function

In this section we consider variants of P systems using a so-called *fairness function* for choosing a multiset of rules out of the set of all multisets of rules applicable to a configuration.

3.1 The General Idea of a Fairness Function in P Systems

Take any (standard) variant of P systems and any (standard) derivation mode. The application of a multiset of rules in addition can be guided by a function computed based on specific features of the underlying configuration and of the multisets of rules applicable to this configuration. The choice of the multiset of rules to be applied then depends on the function values computed for all the applicable multisets of rules.

Therefore, in general we extend the model of a hierarchical P system to the model of a *hierarchical P system with fairness function* (*fair P system* for short)

$$\Pi = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_i, h_o, f),$$

where f is the fairness function defined for any configuration C of Π, the corresponding set $Appl_\delta(\Pi, C)$ of multisets of rules from Π applicable to C in the given derivation mode δ, and any multiset of rules $R \in Appl_\delta(\Pi, C)$. We then use the values $f(C, Appl_\delta(\Pi, C), R)$ for all $R \in Appl_\delta(\Pi, C)$ to choose a multiset $R' \in Appl_\delta(\Pi, C)$ of rules to be applied to the underlying configuration C. A standard option for choosing R' is to require it to yield the minimal value for the fairness function, i.e., we require $f(C, Appl_\delta(\Pi, C), R') \leq f(C, Appl_\delta(\Pi, C)), R)$

for all $R \in Appl_\delta(\Pi, C)$. As usually the derivation mode δ will be obvious from the context, we often shall omit it.

The fairness function may be independent from the underlying configuration, i.e., we may write $f(Appl(\Pi, C), R)$ only; in the simplest case, f is even independent from $Appl(\Pi, C)$, hence, in this case we only write $f(R)$.

Fair or Unfair

One may argue that it is fair to use rules in such a way that each rule should be applied if possible and, moreover, all rules should be applied in a somehow *balanced* way. Hence, a fairness function for applicable multisets should compute the best value for those multisets of rules fulfilling these guidelines.

On the other hand, we may choose the multiset of rules to be applied in such a way that it is the *unfairest* one. In this sense, let us consider the following *unfair example*.

Example 2. Consider the P system $\Pi_1 = (\{a, b\}, \{b\}, [\,_1\,]_1, a, R_1, 1)$ with the rule set $R_1 = \{1 : a \to aa, 2 : a \to b\}$ as considered in Example 1 together with the fairness function f_2 defined as follows: if a rule is applied n times then it contributes to the function value of the fairness function f_2 for the multiset of rules with 2^{-n}. The total value for $f_2(R)$ for a multiset of rules R containing k copies of rule $1 : a \to aa$ and m copies of rule $2 : a \to b$ then is the sum $2^{-k} + 2^{-m}$. The resulting fair P system $\Pi_2 = (\{a, b\}, \{b\}, [\,_1\,]_1, a, R_1, 1, f_2)$ is depicted in Figure 2; we observe that it can also be written as (Π_1, f_2).

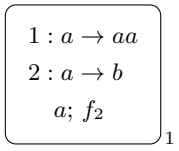

Fig. 2. The P system Π_2

In this fair P system (or in this case we might also call it maximally unfair) with one membrane working in the maximally parallel way, we again start with the axiom a and use the two rules $1 : a \to aa$ and $2 : a \to b$. If we apply only one of these rules to all m objects a, then the function value is 2^{-m} and is minimal compared to the function values computed for a mixed multiset of rules using both rules at least once.

Starting with the axiom a we use the rule $1 : a \to aa$ in the maximal way k times thus obtaining 2^k symbols a. Then in the last step, for all a we use the rule $2 : a \to b$ thus obtaining 2^k symbols b. We cannot mix the two rules in one of the derivation steps as only the clean use of exactly one of them yields the minimal value for the fairness function.

We observe that the effect is similar to that of controlling the application of rules by the well-known control mechanism called label selection, e.g., see [4], where either the rule with label 1 or the rule with label 2 has to be chosen. We will return to this model in Section 4.3. □

The following weird example shows that the fairness function should be chosen from a suitable class of (at least recursive) functions, as otherwise the whole computing power comes from the fairness function:

Example 3. Take the fair P system Π_3 with one membrane working in the maximally parallel way, starting with the axiom a and using the three rules $1: \ a \to aa$, $2: \ a \to a$, and $3: \ a \to b$, see Figure 3.

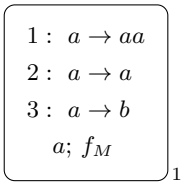

Fig. 3. The P system Π_3

Moreover, let $M \subset \mathbb{N}_+$, i.e., an arbitrary set of positive natural numbers. The fairness function f_M on multisets of rules over these three rules and a configuration containing m symbols a is defined as follows: For any multiset of rules R containing copies of the rules $1: \ a \to aa$, $2: \ a \to a$, and $3: \ a \to b$,

- $f(R) = 0$ if R only contains m copies of rule 3 and $m \in M$,
- $f(R) = 0$ if R only contains exactly one copy of rule 1 and the rest are copies of rule 2,
- $f(R) = 1$ for any other applicable multiset of rules.

Again the choice is made by applying only multisets of rules which yield the minimal value $f(R) = 0$. If we use rule $1: \ a \to aa$ once and rule $2: \ a \to a$ for the rest, this increases the number of symbols a in the skin membrane by one. Thus, in $m-1$ steps we get m symbols a. If m is in M, we now may use rule $3: \ a \to b$ for all symbols a, thus obtaining m symbols b, and the system halts. In that way, the system generates exactly $\{b^m \mid m \in M\}$.

To make this example a little bit less weird, we may only allow computable sets M. Still, the whole computing power is in the fairness function f_M alone, with f_M only depending on the multiset of rules. □

We now again return to Example 2 and illustrate how the same result can be obtained by using another fairness function in the standard *unfair* mode using the multsets of rules with minimal fairness value; on the other hand, we will also show what happens if we try to be *fair* and use the rules in a *balanced* way.

Example 4. Consider the P system $\Pi_1 = (\{a, b\}, \{b\}, [\,_1\,]_1, a, R_1, 1)$ with the rule set $R_1 = \{1 : a \to aa, 2 : a \to b\}$ as considered in Example 1 together with the fairness function $f_4(R)$ for any multiset R of rules defined as follows: consider $f_4(R) = |str(R)|$, i.e., $f_4(R)$ is the number of different strings representing the multiset R. The resulting fair P system $\Pi_4 = (\Pi_1, f_4) = (\{a, b\}, \{b\}, [\,_1\,]_1, a, R_1, 1, f_4)$ is depicted in Figure 4.

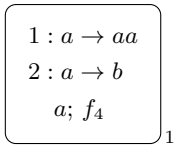

Fig. 4. The P system Π_4

With the standard selection of multisets of rules to be applied by choosing those with the minimal value of the fairness function, we obtain the same result for the set of multisets generated by Π_4, i.e., $\{a^{2^n} \mid n \in \mathbb{N}\}$, because only the pure multisets of rules R containing only copies of rule 1 or only copies of rule 2 yield $f(R) = 1$, whereas any mixed multiset of rules containing both rules at least once yields a bigger value.

On the other hand, if we try to be *fair* and use both rules in a balanced way, i.e., by choosing those multisets of rules yielding the maximum values of f_4, then the generated set is the singleton $\{b\}$, which can be generated in one step from the axiom a by using rule $2 : a \to b$. Any other derivation starting with using rule $1 : a \to aa$ will not yield any result due to running into an infinite computation without any chance to halt: as soon as aa has been generated, only once the rule $1 : a \to aa$ and once the rule $2 : a \to b$ can be used as only this combination of rules yields $f_4(\langle 1, 2\rangle) = |\{12, 21\}| = 2 > 1 = f_4(\langle 1, 1\rangle) = f_4(\langle 2, 2\rangle)$ (we here use the brackets $\langle, \rangle$ to describe a multiset). □

The problem with halting observed in the example above when using only non-cooperative rules seems to be an inherent one when using a *fair* (balanced) selection of multisets of rules. These variants may deserve further investigations in the future, but in this paper we will restrict ourselves to the standard (*maximally unfair*) selection of multisets of rules to be applied as in the previous examples.

4 First Results

In this section, we show three general results. The first one describes how priorities can be simulated by a suitable fairness function in P systems of any kind working in the sequential mode. The second one exhibits how P systems with energy control, see [1], can be simulated by suitable fair P systems for any arbitrary derivation

mode. Finally we show how P systems with rule label control, see [4], can be simulated by suitable fair P systems for any arbitrary derivation mode.

4.1 Simulating Priorities in the Sequential Derivation Mode

In the sequential derivation mode, exactly one rule is applied in every derivation step of the P system Π. Given a configuration C and the set of applicable rules $Appl(\Pi, C)$ not taking into account a given priority relation $<$ on the rules, we define the fairness function to yield 0 for each rule in $Appl(\Pi, C)$ for which no rule in $Appl(\Pi, C)$ with higher priority exists, and 1 otherwise. Thus, only a rule with highest priority can be applied. More formally, this result now is proved for any kind of P systems working in the sequential derivation mode:

Theorem 1. *Let $(\Pi, <)$ be a P system of any kind with the priority relation $<$ on its rules and working in the sequential derivation mode. Then there exists a fair P system (Π, f) with fairness function f simulating the computations in $(\Pi, <)$ selecting the multisets of rules with minimal values.*

Proof. First we observe that the main ingredient Π is exactly the same in both $(\Pi, <)$ and (Π, f), i.e., we only replace the priority relation $<$ by the fairness function f. As already outlined above, for any configuration C of Π we now define f for any rule r as follows (we point out that here the fairness function not only depends on $\{r\}$, but also on $Appl(\Pi, C)$):

- $f(Appl(\Pi, C), \{r\}) = 0$ if and only if there exists no rule $r' \in Appl(\Pi, C)$ such that $r < r'$, and
- $f(Appl(\Pi, C), \{r\}) = 1$ if and only if there exists a rule $r' \in Appl(\Pi, C)$ such that $r < r'$.

If we now define the task of f as choosing only those rules with minimal value, i.e., a rule r can be applied to configuration C if and only if $f(Appl(\Pi, C), \{r\}) = 0$, then we obtain the desired result. □

4.2 Simulating Energy Control

Recently we have considered P systems where a specific amount of energy is assigned to each rule, see [1]. There, only those multisets of rules are applied which use the minimal amount of energy. In a similar way the amount of energy coming up with a multiset of rules can be seen as the value of the fairness function. The minimal amount of energy then exactly corresponds with the minimal fairness.

In this paper, from the two variants of energy-controlled P systems we only consider the one where the energy is directly assigned to the rules. This variant of P systems is called a *rule energy-controlled* P system. The multisets or sets of rules to be applied to a given configuration must fulfill the condition of yielding the minimal amount of energy.

Formally, in a rule energy-controlled P system the rules are of the form (p, v) where p is a rule of a specific type like cooperative or non-cooperative and v is an integer energy value. The total energy value of a mutiset of rules can be defined in different ways, but in the following we will assume it to simply be the sum of energy values of the rules in the multiset and denote this function computing the energy value of a multiset of rules in this way by σ.

Theorem 2. *Let (Π, σ) be a rule energy-controlled P system working in any derivation mode, using any kind of rules and using the sum function σ for computing the energy value of a multiset of rules. Then there exists a fair P system (Π', f) with fairness function f simulating the computations in (Π, σ) with f selecting the multisets of rules with minimal values.*

Proof. By definition, in the rule energy-controlled P system (Π, σ) a multiset of rules can be applied to given configuration only if the application of σ yields the minimal value in $\mathbb{Z}$. The fair P system (Π', f) with fairness function f now is constructed from (Π, σ) by replacing any rule with energy (p, v) by the rule p itself, but on the other hand defining the fairness function f for a multiset of rules to take v as the value assigned to the rule p having been obtained from (p, v). By summing up these values for the whole multiset and selecting only those multisets of rules applicable to a given configuration in the given derivation mode which have minimal values, f fulfills the same task in (Π', f) as σ does in (Π, σ). Hence, in any derivation mode, (Π', f) simulates exactly step by step the derivations in (Π, σ), obviously yielding the same computation results. □

4.3 Simulating Label Selection

In P systems with label selection only rules belonging to one of the predefined subsets of rules can be applied to a given configuration, see [4].

For all the variants of P systems defined in Section 2, we may consider to label all the rules in the sets $R_1, \ldots, R_m$ in a one-to-one manner by labels from a set H and to take a set W containing subsets of H. Then a *P system with label selection* is a construct

$$\Pi^{ls} = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_i, h_o, H, W),$$

where $\Pi = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_i, h_o)$ is a P system as in Section 2, H is a set of labels for the rules in the sets $R_1, \ldots, R_m$, and $W \subseteq 2^H$. In any transition step in Π^{ls} we first select a set of labels $U \in W$ and then apply a non-empty multiset R of rules applicable in the given derivation mode restricted to rules with labels in U.

The following proof exhibits how the fairness function can also be used to capture the underlying derivation mode.

Theorem 3. *Let (Π, H, W) be a P system with label selection using any kind of rules in any kind of derivation mode. Then there exists a fair P system (Π', f) with fairness function f simulating the computations in (Π, H, W) with f selecting the multisets of rules with minimal values.*

Proof. By definition, in the P system (Π, H, W) with label selection a multiset of rules can be applied to given configuration only if all the rules have labels in a selected set of labels $U \in W$. We now consider the set of all multisets of rules applicable to a configuration C, denoted by $Appl_{asyn}(\Pi, C)$, as it corresponds to the asynchronous derivation mode (abbreviated $asyn$); from those we select all R which obey to the label selection criterion, i.e., there exists a $U \in W$ such that the labels of all rules in R belong to U, and then only take those which also fulfill the criteria of the given derivation mode restricted to rules with labels from U.

Hence we define (Π', f) by taking $\Pi' = \Pi$ and, for any derivation mode δ, f_δ for any multiset of rules $R \in Appl_{asyn}(\Pi, C)$ as follows:

- $f_\delta(C, Appl_{asyn}(\Pi, C), R) = 0$ if there exists a $U \in W$ such that the labels of all rules in R belong to U, and, moreover, $R \in Appl_\delta(\Pi_U, C)$, where Π_U is the restricted version of Π only containing rules with labels in U, as well as
- $f_\delta(C, Appl_{asyn}(\Pi, C), R) = 1$ otherwise.

According to our standard selection criterion, we choose only those multisets of rules where the fairness function yields the minimal value 0, i.e., those R such that there exists a $U \in W$ such that the labels of all rules in R belong to U and R is applicable according to the underlying derivation mode with rules restricted to those having a label in U, which exactly mimicks the way of choosing R in (Π, H, W). Therefore, in any derivation mode δ, (Π', f_δ) simulates exactly step by step the derivations in (Π, H, W), obviously yielding the same computation results. □

5 Conclusions and Future Research

In this article, we introduced and partially studied P systems with the application of rules in each step being controlled by a function on the applicable multisets of rules.

We have given several examples exhibiting the power of using suitable fairness functions. Moreover, we have shown how priorities can be simulated by a suitable fairness function in P systems of any kind working in the sequential mode as well as how P systems with energy control or label selection can be simulated by fair P systems with a suitable fairness function for any derivation mode.

Yet with all these examples and results we have just given a glimpse on what could be investigated in the future for P systems in connection with fairness functions:

- consider other variants of hierarchical P systems working in different derivation modes, e.g., also taking into consideration the set derivation modes;
- extend the notion of *fair* to tissue P systems, i.e., P systems on an arbitrary graph structure;
- extend the notion of fair to P systems with active membranes, there probably also controlling the division of membranes;
- investigate the effect of selecting the multiset of rules to be applied to a given configuration by other criteria than just taking those yielding the minimal values for the fairness function;
- consider other variants of fairness functions, either less powerful or taking into account other features of $Appl(\Pi, C)$ and/or the multiset of rules R;
- investigate the effect of selecting the multiset to be applied to a given configuration by requiring it to contain a balanced (really *fair*) amount of copies of each applicable rule;
- show similar simulation results with suitable fairness functions as in Section 4 for other control mechanisms used in the area of P systems;
- ...

References

1. Artiom Alhazov, Rudolf Freund, and Sergiu Ivanov. Variants of energy-controlled P systems. In *Proceedings NIT 2016*, 2016.
2. Rudolf Freund. P systems working in the sequential mode on arrays and strings. In Cristian Calude, Elena Calude, and Michael J. Dinneen, editors, *Developments in Language Theory, 8th International Conference, DLT 2004, Auckland, New Zealand, December 13-17, 2004, Proceedings*, volume 3340 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2004.
3. Rudolf Freund, Alberto Leporati, Giancarlo Mauri, Antonio E. Porreca, Sergey Verlan, and Zandron. Flattening in (tissue) P systems. In Artiom Alhazov, Svetlana Cojocaru, Marian Gheorghe, Yurii Rogozhin, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, volume 8340 of *Lecture Notes in Computer Science*, pages 173–188. Springer, 2014.
4. Rudolf Freund, Marion Oswald, and Gheorghe Păun. Catalytic and purely catalytic P systems and P automata: Control mechanisms for obtaining computational completeness. *Fundamenta Informaticae*, 136(1-2):59–84, 2015.
5. Gheorghe Păun. Computing with Membranes. *Journal of Computer and System Sciences*, 61:108–143, 1998.
6. Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.
7. Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, 3 volumes*. Springer, New York, NY, USA, 1997.
8. Bulletin of the International Membrane Computing Society (IMCS). `http://membranecomputing.net/IMCSBulletin/index.php`.
9. The P Systems Website. `http://ppage.psystems.eu/`.