

(In Search of) Probabilistic P Systems¹

Adam OBTUŁOWICZ

Institute of Mathematics of the Polish Academy of Sciences
Śniadeckich 8, PO Box 137, 00-950 Warsaw, Poland
E-mail: adamo@impan.gov.pl

Gheorghe PĂUN

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 70700 București, Romania, and
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: gpaun@imar.ro, gp@astor.urv.es

Abstract. The aim of this paper is to (preliminarily) discuss various ways of introducing probabilities in membrane systems. We briefly present both ideas already circulated in the literature and new proposals, trying to have a systematic overview of possibilities of associating probabilities with the ingredients of a membrane system: with single objects, with multiplicities of objects (hence with the multisets), with the rules (depending or not on the previous applied rule), with the communication targets. For a certain mode of using the probabilities associated with the evolution rules (in string-object P systems) we obtain the computational universality.

Keywords: Turing computability, Bio-chemistry, Membrane Computing, Probability

1 Introduction

Membrane computing is a branch of natural computing aiming to abstract computing ideas/models from the structure and the functioning of living cells, an area which should be seen in a close relation with DNA computing, with molecular computing in general: while molecular computing deals with computing-like processes which takes place, for instance, at the genetic level, membrane computing is an extension to the level of the cell. More abstractly stated, while molecular computing deals with single-compartment processes, membrane computing proposes and investigates multi-compartmental computing models, similar to the cell architecture. In this framework, important issues can be addressed, such as distribution, cooperation, synchronization, communication. For the readers convenience, in the next section we will briefly present the basic notions and (types of) results of the membrane computing area.

Almost everything which has been done up to now in membrane computing is done in terms of deterministic crisp mathematics: without probabilities, without fuzzy or rough set theory elements. However, the model itself is non-deterministic in the computer science sense: the objects from a given compartment can be processed by several rules, which are chosen in a non-deterministic manner – this corresponds to equal probabilities assigned to the rules which can be used at a given time. Note the important detail that the probabilities implicitly associated with non-determinism in the above sense are “evaluated” in a dynamic manner, *for the rules*

¹Work done in the framework of the Contract No ICA1-CT-2000-70024 between IMPAN, Warsaw, Poland, and the European Community

which can be applied at a given moment in a given compartment. The rules which cannot be applied are supposed to have probability zero, those which can be applied have equal non-zero probabilities.

This is not exactly what we meet in reality, *in vivo* or *in vitro* equally. Some reactions are more active than others, and this can be valid in general, not necessarily depending on the local conditions (the “buffer”). On the other hand, the recent “history” of a compartment of the cell can determine the next reactions to take place in that compartment, hence a sort of Markovian dependence among reactions could exist. Very convincing and well motivated/realistic proofs of the usefulness of the probabilistic approach when modeling various biochemical processes can be found in [Bower, Bolouri, Eds., 2001].

In terms of membrane systems (see also the next section), non-determinism – hence probabilities – can appear also when transferring (we say “communicating”) objects from a compartment to another one, and this is expected to also correspond to a biological fact. This brings into discussion the idea of probabilities associated with objects – with their localization into the cell/system. Imagine, for instance, the lab operation of introducing certain molecules in a cell compartment by means of a micro-pipe. Because of the reduced size of the target, we can only have an estimation of the place where the molecules arrive after the operation. This means probabilistic expectations about the place where each molecule is, as well as probabilistic estimations of the number of copies of each molecule in neighboring compartments.

In short, both *in vivo* and *in vitro* we encounter probabilities (not necessarily because *God plays dices* – Einstein’s terms – but because of our knowledge and technological limitations), hence it is natural to consider probabilities also in membrane systems. This is both a mathematical challenge and a possible starting point in order to make membrane systems relevant for biologists. Membrane computing is biologically inspired but mainly motivated by mathematical and computer science goals (getting computing models which are powerful, efficient, elegant – not necessarily realistic). In view of the lack of global formal models of the cell, it is however important to try to keep the membrane systems theory as close as possible to the biological reality and to try to return to biologists some relevant results. Considering probabilities can be a first step in this respect, followed by a mathematical step and, quite standard and promising, a step based on computer simulations.

The idea of adding probabilities to P systems can be also placed in a broader perspective, that of the interplay of *qualitative* and *quantitative* models in biology. This is a much debated topic and we do not want to enter into details, but we only quote some lines from an illustrative paragraph from the Foreword (by J.C. Wooley) of [Bower, Bolouri, Eds., 2001]: “Throughout this same quarter century of remarkable progress, biologists have distrusted modeling, theory, and analytical and mathematical analysis. Researchers have focused on simple systems, [...] and whenever possible, asked questions believed to yield only binary answers, not quantitative ones. Completion and analysis of the DNA sequence of the first model organism, *H. influenza*, marked the end of our innocence; similarly, entry into the contemporary high throughput era, initiated by the human genome project, unambiguously demonstrated the limits of qualitative biology and introduced the vision of quantitative biology at a system level.” Of course, we are not advocating for one of the two paradigms, for the quantitative or for the qualitative one, there are well-known models, even theories, of a recognized impact (also) in biology and which keep a good balance of quantitative and qualitative aspects – let us mention only the R. Thom’s catastrophe theory and Mandebrod’s fractals. Actually, also membrane computing is a “hybrid” approach, a bridge between qualitative and quantitative: the style is that of rewriting devices in formal language theory (hence of a qualitative type), but dealing with multisets of objects, with vectors of numbers, hence with quantities. Sometimes, the quantitative features are enhanced

– see, for instance, the classes of P systems where one also considers energy associated with the rules or with the objects, in form of numerical values ([Freund, 2002], [Frisco, Ji, 2002], etc.). Probabilistic P systems are a further step towards a quantitative “stage” of membrane computing, which coincides with a somewhat visible change of the focus of interest in the domain: from computer science, to biological modeling. One illustration of this shift of interest (better said: enlargement of preoccupations) is the paper [Bernardini, Manca, 2002], approaching P systems from the dynamic systems perspective, addressing types of problems completely new for the domain. We strongly believe that this direction of research, dealing with the “life” of a P system, not with its Turing-like computing properties, is worth engaging and will prove to be very fruitful.

In this paper we only discuss various possibilities of introducing probabilities in membrane systems, at the level of objects or of rules, with various possibilities in each case, without going into mathematical investigations (only one theoretical result is given, because of its relevance for the qualitative–quantitative distinction: universality for a certain mode of using the probabilities associated with the rules of a rewriting P system) and without discussing possible ways of using these ideas for constructing computer programs for abstract or real-life case-studies. However, we expect continuations from both these points of view (especially computer simulations), having in mind the importance of the cell simulation for biochemists.

2 A Glimpse to Membrane Computing

Membrane computing is a rather young area – it was initiated in [Păun, 2000] (the paper was circulated on web at the end of 1998) – but it is rather developed from a mathematical point of view. We will recall here only the most basic ideas and results, and we refer the reader to the web page <http://psystems.disco.unimib.it> and to the monograph [Păun, 2002] for further details, in particular, for bibliographical information. Friendly overviews of the field can also be found in [Păun, 2001] and [Păun, Rozenberg, 2002].

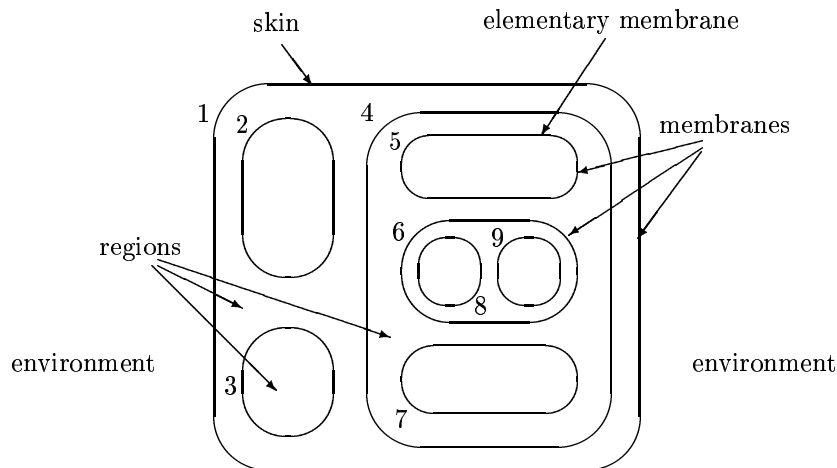


Figure 1: A membrane structure

Membrane computing starts from the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. Abstracting from the biochemical details, one gets *membrane systems* (called also P systems), which, roughly speaking, consist of a cell-like *membrane structure* in the compartments of which one places

multisets of objects and (sets of) evolution rules.

The *membrane structure* is a hierarchical arrangement of membranes (understood as three dimensional vesicles), embedded in a *skin* membrane, the one which separates the system from its *environment*. A membrane without any membrane inside is called *elementary*. Each membrane defines a *region*. For an elementary membrane this is the space enclosed by it, while the region of a non-elementary membrane is the space in-between the membrane and the membranes directly included in it. Figure 1 illustrates these notions. We label membranes (by positive integers in Figure 1) in order to be able to address them when we program computations by membrane systems. Since each region is delimited “from the outside” by a unique membrane, we will use the labels of membranes to also identify (label) the regions they delimit.

Each region contains a multiset of *objects* (they correspond to the molecules present in the cell compartments, from ions to large macromolecules), and a set of (*evolution*) *rules* (they correspond to chemical reactions possible in each compartment). In the basic variant of P systems we discuss here, the objects are represented by symbols from a given alphabet (they are “atoms”, without any structure); there are variants of P systems where the objects are represented by strings of symbols from a given alphabet (this corresponds to the case when the structure of chemicals is relevant).

It is important to stress the fact that both the objects and the rules are localized, they belong to specific compartments defined by the membranes of the system.

Typically, an evolution rule (from a region r) is of the form $ca \rightarrow cb_{in}d_{out}d_{here}$, and it “says” that a copy of the object a , in the presence of a copy of the *catalyst* c (this is an object which is never modified, it only assists the evolution of other objects), is replaced by a copy of the object b and two copies of the object d . Moreover, the copy of b has to enter “immediately” one of the membranes directly placed in the region r (one of the membranes which delimit region r “from inside”), one copy of object d is sent out through the membrane of region r , and one copy of d remains in region r . Note that the considered evolution rule can be applied in the region r only if this region includes at least one inner membrane, so that the command “in” can be executed. The transfer of objects from a region to another region is called *communication*, and it corresponds to the passage of molecules through membranes, through protein channels, in a passive or active manner.

The application of evolution rules (hence the system evolution) is performed in a synchronous, parallel, and non-deterministic manner: at each moment (the time is marked by the same clock for all regions), all objects which can evolve by the rules available in that compartment should do it; when several rules can be applied to the same object, then the rule to be used is non-deterministically chosen. Moreover, the membranes can be dissolved (broken), divided, created. An evolution of a membrane system is a *computation*; we consider as successful only the *halting computations*, those which reach a configuration where no further rule can be applied. A *result* is associated with a successful computation. In the case of P systems with symbol-objects, the typical result of a halting computation is the number of objects sent outside the system during the computation, in the case of P systems working with string-objects the result of a computation consists of the strings sent out of the system during the computation.

Many modifications/extensions of this very basic model are discussed in the literature. Most of them are computationally complete, i.e., equal in power to Turing machines. If an exponential workspace can be created (in polynomial time), by dividing membranes, or by replicating string-objects, or by creating membranes from objects which can be replicated, then polynomial time solutions to NP-complete problems can be obtained. Therefore, many classes of P systems have the two basic good properties of a computing model: they are *computationally universal* and, also, *computationally efficient*.

Note that we have several sources of non-determinism in a P system as sketched above: several rules can involve the same object, a given object can be present in several copies which can evolve by the same rule, a target command *in* can point to several inner membranes at the same time; when we have string-objects (processed by rewriting rules), the rules can be applied to several occurrences of the same symbol in a string, or we can have several applicable rules, etc.

There are several ways to decrease the non-determinism, hence to control the use of rules. We only mention three possibilities: (1) to consider a priority relation among rules, in the form of a partial order relation, and to use a rule in a given step only if no rule of a higher priority is applicable; (2) to control the membrane permeability, by making membranes *impermeable* (no object can pass through such a membrane), or returning them to normal permeability, or even *dissolving* membranes (in such a case, all objects remain free in the upper membrane; the skin membrane is never dissolved); (3) to indicate targets for the objects to be communicated to inner membranes, that is, using commands in_j instead of *in* (with the meaning “got to membrane j ”), or at least associating electrical charges to objects and to membranes, so that the objects are introduced into membranes of opposite polarization.

In the above discussion, the symbol-objects were processed by multiset rewriting-like rules (some objects are transformed into other objects, which have associated targets). Coming closer to the trans-membrane transfer of molecules – and also very important from a computability point of view – we can consider purely communicative systems, based on the two types of coupled transport known in the biology of membranes: *symport* and *antiport* [Alberts et al., 1994], [Ardelean, 2002]. Symport refers to the transport where two molecules pass together through a membrane in the same direction, antiport refers to the transport where two molecules pass through a membrane simultaneously, but in opposite directions. In terms of P systems, we can consider object processing rules of the following forms: a symport rule (associated with a membrane i) is of the form (ab, in) or (ab, out) , stating that the objects a and b enter, respectively exit together membrane i , while an antiport rule is of the form $(a, out; b, in)$, stating that, simultaneously, a exits and b enters membrane i . From a mathematical point of view, it is natural to consider generalizations of such rules, moving at the same time several objects through a membrane. For instance, the general form of an antiport rule is $(x, out; y, in)$, where x and y are arbitrary (but non-empty) multisets of objects.

What is very attractive in the case of symport/antiport is that no object is created or destroyed, only the place of the objects is changed (this also means that the conservation law is observed – which does not necessarily happen in other classes of P systems). Also, in this case the environment is an active participant to the computation, holding as many copies of each object as necessary, and involved in a two-way communication with the skin region of the system.

Interesting enough, P systems with symport/antiport rules are capable of universal computations; otherwise stated, communication alone (in the framework of P systems with symport/antiport) entails computational universality.

3 Probabilities at the Level of Objects

Let us start from the situation mentioned in the Introduction, where we do not precisely know the region where an object is placed, but we only have some estimations about that. Actually, because we work with multisets of objects, hence with numbers of copies associated with each object, we have to distinguish two cases: (1) estimating the position (the region) of each copy of

an object, individually, and (2) estimating the “population” of objects in a given region, their multiplicity.

In the first case, for each object a from the given alphabet V of possible objects, we have to consider copies, for instance, indexed with natural numbers: a_1, a_2, \dots . When tracing the position of a copy a_i of a we have to distinguish it from other copies, a_j with $j \neq i$, although for an evolution rule involving the object a the two copies of a are identical (the rules are given in terms of object-names, of “prototypes of objects”, not in terms of instances).

Let us consider a membrane structure μ (given, as usual, as a string of correctly matching labeled parentheses, embedded in the external pair of parentheses associated with the skin membrane; for instance, the membrane structure from Figure 1 is described by the expression $[_1[_2[_3]_3]_4[_5]_5]_6[_8]_8]_9]_9[_6]_6]_7]_7]_4]_4]_1$). Assume that μ contains m membranes, labeled in an injective way with $1, 2, \dots, m$, and also assign label 0 to the environment. Consider an object $a \in V$ and an instance a_j of it, which we know for sure to exist somewhere, in the system or outside it. When having only estimations about the place of a_i , we can consider probabilities $p_i(a_j)$ that a_j is present in region i , $0 \leq i \leq m$. Because a_j exists for sure, we must have $\sum_{i=0}^m p_i(a_j) = 1$.

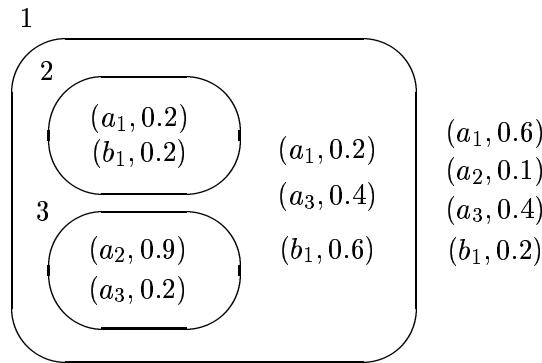


Figure 2: Probabilities associated with objects

For instance, take $\mu = [_1[_2[_3]_3]_3]_1$, as well as two objects a and b for which we know for sure that the system with the membrane structure μ contains three copies of a and one copy of b . Possible estimations of the probability to have each of these objects in the compartments defined by μ can be as illustrated by Figure 2, where each copy of an object is accompanied by the probability to have that copy in the respective region.

Although it is easy to associate probabilities to objects (to their places in a membrane structure), it is not similarly easy to handle these probabilities. A basic question is the following: should we continue to work with such a probabilistic distribution of objects (then the next question is *how?*), or we have to try to improve the knowledge about the objects localization, by performing certain *experiments* (and then the next question is *what kind of experiments?*)?

For instance, deciding to use a rule in a compartment could be considered like a measurement in quantum physics, removing all the uncertainty about the involved objects: just because we use the rule, we settle to have the objects in that compartment. For instance, using the rule $ab \rightarrow c_{tar_1} b_{tar_2} b_{tar_3}$ in region 1 of the system from Figure 2 will completely determine the position of b_1 , as well as of one of a_1 and a_3 , the one used by the rule. Assume that we evolve a_3 and b_1 by this rule. This means that a_3 was settled to be in region 1, hence it is not in region 0 (the environment) or in region 3, and the corresponding probabilities are set to zero. The same with the probability to have b_1 in membrane 2 or in the environment. The uncertainty about a_1 and

a_2 remains. New probabilities can appear for the newly produced objects, c, b, b , for instance, in the case when they have associated the target in . Assume that the rule contains the symbol c_{in} . Because c can enter either membrane 2 or membrane 3, we may put $(c_1, 0.5)$ in each of them.

Of course, at the same time with the rule $ab \rightarrow c_{in}b_{tar_2}b_{tar_3}$ in region 1, for evolving a_3 and b_1 , we cannot use any rule in another region which involves the same copies of a and b . This could lead to an interesting interplay of rules, and points out again to a quantum physics notion, that of *entanglement*; here, several “particles” $(a_j, p_i(a_j))$ are “entangled” in the sense that when one of them is used by a rule, none of the other “particles” with the same a_j on the first position can be used by a rule; moreover, all these “particles” disappear when one of them is used.

In the above way of using the object–probability pairs, the value of the probability does not matter, the only important fact is whether it is zero or not (a qualitative information). A different situation appears when using symport/antiport rules for evolving the objects. By using a rule, the (copies of) objects only change their places, hence they can carry with them the associated probability. Actually, when using a rule, we may assume that it is used in an extent equal to the minimal value among the probabilities of the involved objects. Then, the probability of the evolved objects is decreased with this minimal value, and the objects transferred through membranes carry with them this minimal probability.

Let us illustrate this idea for the case from Figure 2; a formalization remains as a task for the reader interested in mathematical developments.

Assume that we have the antiport rule $(a, out; ab, in)$ associated with membrane 3. We have two copies of a in region 3, $(a_2, 0.9)$ and $(a_3, 0.2)$, one copy of b , $(b_1, 0.6)$, and two copies of a , $(a_1, 0.2)$ and $(a_3, 0.4)$, in region 1. Thus, we have four possibilities of using this rule (the underlined objects are those which give the minimal probability):

$$\begin{array}{ll}
(a_2, 0.9) \text{ out} & \text{and } (a_1, 0.2)(b_1, 0.6) \text{ in} \\
\text{or } (a_2, 0.9) \text{ out} & \text{and } \underline{(a_3, 0.4)}(b_1, 0.6) \text{ in} \\
\text{or } \underline{(a_3, 0.2)} \text{ out} & \text{and } \underline{(a_1, 0.2)}(b_1, 0.6) \text{ in} \\
\text{or } \underline{(a_3, 0.2)} \text{ out} & \text{and } (a_3, 0.4)(b_1, 0.6) \text{ in}
\end{array}$$

After using the rule, we obtain:

	in region 3	in region 1
	$(a_2, 0.7), (a_1, 0.2), (b_1, 0.2)$	$(a_2, 0.2), (b_1, 0.4)$
or	$(a_2, 0.5), (a_3, 0.4), (b_1, 0.4)$	$(a_2, 0.4), (b_1, 0.2)$
or	$(a_1, 0.2), (b_1, 0.2)$	$(a_3, 0.2), (b_1, 0.4)$
or	$(a_3, 0.2), (b_1, 0.2)$	$(a_3, 0.4), (b_1, 0.4)$

The objects which were not affected by the use of the rule were not mentioned. Note that in the last case, $(a_3, 0.2)$ was both sent out of region 3 and reintroduced into region 3, hence its “membership degree” to this region is not changed.

Using the rules as suggested above, we can get transitions in a P system with symport/antiport rules, in the usual manner (at each step, each a_j from $(a_j, p_i(a_j))$ is involved in only one rule, but the rules are applied in parallel, in the maximal manner, to all objects a_j which can evolve, hence the probabilities are recounted after each step, according to all applied rules). As usual, a sequence of transitions constitutes a computation and a computation is considered successful only if it halts. The result of a successful computation is “read” in an *output* membrane, an elementary one specified in advance (the environment contains objects, hence it is “safer” to define the result in an internal way). Specifically, if i_o is the output membrane and

$\lambda \in [0, 1)$ is a given *acceptance threshold*, then the result of a computation is the number of objects a_j such that $p_{i_o}(a_j) > \lambda$ (of course, the probability is computed in the halting configuration of the computation). Let us denote by $N_\lambda(\Pi)$ the set of all numbers computed in this way by a P system Π . By $NOpP_m^\lambda(sym_u, anti_v)$ we denote the family of sets $N_\lambda(\Pi)$ of natural numbers computed by systems Π with at most m membranes, using symport rules $(x, in), (x, out)$ with $|x| \leq u$, and antiport rules $(x, out; y, in)$ with $|x|, |y| \leq v$ (for a string x we denote by $|x|$ its length). When considering systems without probabilities, we write NOP instead of $NOpP$. Moreover, we denote by NRE the family of recursively enumerable sets of natural numbers (the family of sets of numbers which can be computed by Turing machines).

A usual (non-probabilistic) P system with symport/antiport rules can be considered a probabilistic one, taking $p_i(a_j) \in \{0, 1\}$, hence for all $\lambda \in [0, 1)$ we have $NOpP_m^\lambda(sym_u, anti_v) = NRE$ for all parameters m, u, v for which $NOP_m(sym_u, anti_v) = NRE$. Therefore, from the computational power point of view, P systems with probabilities assigned to objects are not very interesting. Such systems can however be of interest when they are considered as dynamic systems, not looking for halting computations, but for possibly infinite evolutions. In such a framework, several questions can be formulated about the “life” of a system, with possible biological meanings: Does a region become overcrowded/void in time? Are there objects which completely exit the system or which reach a state of “maximal diffusion”, where they are present everywhere with (almost) equal probability? Which is the ratio in time between the population of two distinguished objects in a given region? Of course, such questions can be both approached by analytical means and, mainly, by computer simulation.

Let us pass now to a variant of the previous way to estimate the position of objects by means of probabilities. Let us suppose that we have some information about the objects present in a given region, without knowing exactly the number of copies of each object. This should be the case in reality, where we can check whether or not we have, say, calcium ions in a given compartment, but it is unreasonable to suppose that we can count these ions. However, some statistics about the concentration of calcium ions are possible. For instance, we can say that “the probability to have n copies of a given object a in region i is $p_{i,a}(n) \in [0, 1]$ ”. Presumably, for values of n around some specified n_0 the value of $p_{i,a}(n)$ is larger, closer to 1, while for n far from n_0 this value is closer to or even equal to 0.

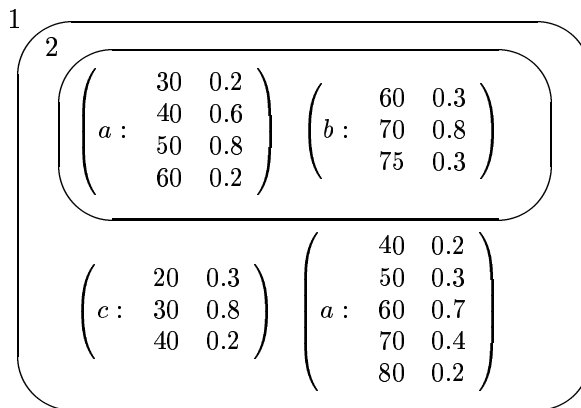


Figure 3: Probabilities associated with multiplicities

This framework looks interesting from several points of view. First, it is again a generalization of the usual P systems, hence its universality holds for all cases where usual systems are universal. Then, we have here a completely new way of specifying a configuration of the system: instead

of multisets, in each region we have sets of pairs of the form $(a, p_{i,a} : \mathbf{N} \rightarrow [0, 1])$. The mappings $p_{i,a}$ should have some minimal properties (the values for two arguments $n, n + 1$ should not be “too different”, a small number of local maxima, value zero for n greater than some threshold, etc.), as well as some relationships with mappings from neighboring regions (in general, the concentration of certain chemicals cannot be very large in one compartment and very small in an adjacent compartment). All these are topics for an experimental investigation. We continue here by pointing out some possibilities/difficulties of using evolution rules in this framework. Let us consider the situation from Figure 3, where we have mentioned only the numbers n for which $p_{i,\alpha}(n)$ is non-zero.

Assume that the antiport rule $(ab, out; c, in)$ is associated with membrane 2. How can we use this rule? Here is one proposal. Consider each possible “state of the world”, that is, combination of probabilities $p_{i,\alpha}(n)$ for $\alpha \in \{a, b, c\}$ as involved in the rule. We have $4 \times 3 \times 3$ combinations. One example: $p_{2,a}(30) = 0.2$, $p_{2,b}(70) = 0.8$, $p_{1,c}(40) = 0.2$. For each combination, take the minimum number of occurrences – in our example, this is 30, for a . The rule is applied this number of times (in our case, 30), moving according to the indications *in*, *out* copies of objects from a region to another one. This means that with probability 0.2 we will have $30 - 30 = 0$ copies of a in region 2, with probability 0.8 we will have $70 - 30 = 40$ copies of b in region 2, and with probability 0.2 we will have $40 - 30 = 10$ copies of c in region 1; with the same probabilities, these objects are moved, a and b to region 1, and c to region 2. Such calculations are performed for all combinations as above. In the end, we will have 9 possible results for each of the four possible multiplicities of a in region 2, and 12 possible results for each of the three multiplicities of b and c in regions 2 and 1, respectively. We list them for the case of a in region 2 (the first two columns are the numbers of copies of a existing in region 2 and their probabilities, the next 9 columns indicate the remaining numbers of copies of a after using the rule, and the last column is the average of these numbers (rounded, in the case of the first row):

30	0.2	10	10	10	0	0	0	0	0	0	3
40	0.6	20	20	20	10	10	10	0	0	0	10
50	0.8	30	30	30	20	20	20	10	10	10	20
60	0.2	40	40	40	30	30	30	20	20	20	30

Therefore, we may say that after the use of the rule $(ab, out; c, in)$ we have $p_{2,a}(3) = 0.2$, $p_{2,a}(10) = 0.6$, $p_{2,a}(20) = 0.8$, and $p_{2,a}(30) = 0.2$. In a similar way are affected the probabilities of multiplicities of b in region 2. Moreover, we will have here 20, 30, 40 copies of c with probabilities 0.3, 0.8, 0.2, respectively. The case of multiplicities of c in region 1 is similar, but for a in region 1 we have a different situation. We bring here 20, 30, or 40 copies of a from membrane 2, with the probabilities 0.2, 0.6, 0.8, 0.2 in each case, depending on the combination chosen as the “state of the world”. Should we continue with the average of these probabilities or with the maximum of them? In both cases, the probabilities of having 20 or 30 occurrences of a in region 1 are equal, and in both cases this value is greater than the estimated probability to have here 40 copies of a . That is, in this way the mapping $p_{1,a}$ gets one further local maximum.

We do not persist into this discussion; the right choice of using the probabilities can be made either on an experimental basis, or using more involved probabilistic tools.

4 Probabilities at the Level of Rules

Associating probabilities with the rules from a given region is a rather natural idea, and it corresponds both to the biochemical reality, where some reactions are more probable than others (more active), and to notions already investigated in (theoretical) computer science,

such as the probabilistic grammars (they are both “old stuff” in formal language theory, see, e.g., [Salomaa, 1969], and, with specific changes, “fashion” in quantum computing, see, e.g., [Moore, Crutchfield, 1997]). Actually, probabilistic P systems with probabilities associated with rules were already considered, both from a “practical” point of view (in computer programs dealing with P systems simulations) and from a theoretical point of view. Of the first type are the approaches from [Nishida, 2002] and [Suzuki et al., 2001]. We do not enter into details concerning the pieces of reality which constitute the topic of the computer-aided investigations, the titles of the two papers are self-explanatory (while the monograph [Păun, 2002] recalls several considerations from these papers). In both [Nishida, 2002] and [Suzuki et al., 2001], the rules from a given region have associated probabilities of application, in the form of subunitary numbers (not necessarily forming a probability space, that is, with the sum of probabilities equal to 1 – although this can be achieved by a suitable normalization). In [Nishida, 2002] these numbers are computed at each step of the simulation, depending on the contents of the compartment. No mathematical investigation is carried in either case, but in both mentioned papers one considers rather realistic case studies, and the conclusions found by computer simulations look sound, even relevant, from a biochemical/biological point of view.

Of a completely different type is the approach from [Madhu, 2002], where one follows the style of [Salomaa, 1969]: the probabilities are assigned to pairs of rules and they describe the expectation to apply one rule after another one. This is somewhat a routine task in the case of P systems with string-objects, with a sequential processing of objects (in each step, one rule is applied to each string – if this is possible, otherwise the string remains unchanged) by means of rewriting rules of the form $X \rightarrow w(\text{tar})$, where $X \rightarrow w$ is a usual context-free rule and tar is one of *here*, *out*, *in*, indicating the region where the string obtained by rewriting should be placed in the next step. More specifically, assume that we have a P system with m regions, and in region i we have the rules $R_i = \{r_{i,1}, \dots, r_{i,k_i}\}$. An initial probability distribution is given, $\delta_i : R_i \rightarrow [0, 1]$, indicating the probability to start a computation with each rule from R_i . Moreover, functions $\varphi_i : R_i \rightarrow [0, 1]^{k_i}$ are given, such that $\varphi_i(r_{i,j}) = (p(r_{i,j}, r_{i,1}), \dots, p(r_{i,j}, r_{i,k_i}))$ are vectors for which we have $\sum_{t=1}^{k_i} p(r_{i,j}, r_{i,t}) = 1$, for all $1 \leq j \leq k_i$. Each $p(r_{i,j}, r_{i,t})$ indicates the probability of using the rule $r_{i,t}$ after the rule $r_{i,j}$ in region i . Using the rules in the usual way, we get computations; in the same way as in the case of probabilistic grammars, each computation has an overall probability, which is obtained by multiplying the probabilities of all rules used. A string sent out during a computation which eventually halts is considered to be generated by the system if it has been obtained at the end of a computation with a probability strictly greater than a threshold λ given in advance, $\lambda \in [0, 1)$.

This kind of selection of the-next-rule-to-be-applied, by means of probabilities, is rather powerful. Let us examine the following system, a simplified version of an example considered in [Madhu, 2002]. We have only one membrane, initially containing the string AB and the following rules:

rule	δ_1	$p(r_i, r_1)$	$p(r_i, r_2)$	$p(r_i, r_3)$	$p(r_i, r_4)$	$p(r_i, r_5)$	$p(r_i, r_6)$
$r_1 : A \rightarrow aA(\text{here})$	1/2	0	0	0	1	0	0
$r_2 : A \rightarrow bA(\text{here})$	1/2	0	0	0	0	1	0
$r_3 : A \rightarrow \varepsilon(\text{here})$	0	0	0	0	0	0	1
$r_4 : B \rightarrow aB(\text{here})$	0	1/3	1/3	1/3	0	0	0
$r_5 : B \rightarrow bB(\text{here})$	0	1/3	1/3	1/3	0	0	0
$r_6 : B \rightarrow \varepsilon(\text{out})$	0	0	0	0	0	0	1

The reader can easily check that with the threshold $\lambda = 0$ this system generates the language $\{ww \mid w \in \{a, b\}^+\}$, which is not a context-free one (note that one-membrane rewriting P systems

– without probabilities – generate only context-free languages). Indeed, we can start with any of the first two rules, and each of these rules is precisely followed by r_4 or r_5 , respectively; in turn, r_4 and r_5 can be followed by any of the first three rules; rule r_6 can be used only after using r_3 , and the string obtained in this way, of the form ww , for some non-empty string w over the alphabet $\{a, b\}$, is sent out of the system.

Thus, it is not very surprising that the probabilistic P systems of degree 1 were proven in [Madhu, 2002] to characterize the power of programmed (hence also matrix) grammars without appearance checking. (We refer to [Dassow, Păun, 1989] for the regulated rewriting area of formal language theory.)

This way of using the probabilities associated with rules is not very satisfactory: the only fact which matters is whether or not a rule has a non-zero probability (hence a qualitative information), the actual value of the probability is ignored. What about taking a quantitative position and take care of the rule “reactivity”, as expressed by the probability? This means to use the rule which has the maximal probability among the applicable rules (when several applicable rules have the same maximal probability we non-deterministically choose one of them). This introduces a priority among the applicable rules, which both corresponds to a biochemical reality and also adds power to our systems.

Actually, *P systems with only one membrane with the probabilities assigned to rules used in the above mentioned way are computationally universal, they can generate all recursively enumerable languages.* The technical formulation and the proof of this result are given in the Appendix.

This result can be interpreted as a somewhat surprising illustration of the difference between the qualitative and the quantitative models – in our framework indicating a clear advantage of the quantitative standpoint: while the qualitative use of probabilities leads to a computing device having the competence of programmed grammars, the quantitative use provides computational universality.

This can be also seen as a particular interpretation/use of the probability concept and a new type of reasoning with it, in a way which is not “classic” from a probabilistic point of view: we do not deal with statistic reasoning, but with a way to control dynamic processes by means of probabilities. We think that this perspective is fruitful and it deserves to be further explored.

A different approach is considered in [Obtulowicz, 2002], where one works with symbol-object P systems. After proving that such systems can simulate Petri nets (actually, a P system is seen as a generalization of a Petri net, such that applications of evolution rules coincide with firings of transitions), one follows [Marsan, 1990], and one associates probabilities of delaying the use of rules. The greatest the delay in applying a rule, the smallest the probability to use it in a given step, hence we have a similar kind of systems as in [Nishida, 2002] and [Suzuki et al., 2001], where probabilities of applying rules were considered.

Which is the computing power – and which is the practical usefulness – of the kind of probabilistic P systems considered in [Obtulowicz, 2002] remains to be investigated. We conclude this section by mentioning that in [Obtulowicz, 2002] one also considers randomized P systems, as a possible support for implementing randomized algorithms. The idea is illustrated by discussing an implementation of the Miller-Rabin randomized algorithm for primality testing of integers, see, e.g., [Koblitz, 1998]. The nice result of this approach is that polynomial solutions can be found, with a high enough probability of correctness, by a random search in a subexponential workspace. This should be compared with the usual way of obtaining polynomial solutions to NP-complete problems in the membrane computing area, by using an exponential workspace created, e.g., by membrane division.

5 Probabilities at the Level of Targets

In the Introduction we have pointed out that one further source of non-determinism – hence a place where probabilities can be introduced – appears at the level of target indications. While *here* and *out* uniquely identify the place where the objects are to be sent, an indication *in* is ambiguous in the case when several membranes exist inside the membrane where the rule which has introduced the indication *in* is used. Then, it is just natural to assign probabilities to the possible target membranes. For instance, when we have membranes j_1, \dots, j_k inside membrane i , instead of α_{in} we can consider $\alpha_{(in_{j_1}, p_1), \dots, (in_{j_k}, p_k)}$, with $\sum_{t=1}^k p_t = 1$. The meaning is that the object α will be moved into membrane j_t with the probability p_t , $1 \leq t \leq k$.

Clearly, this is a generalization of both the command *in*, which corresponds to the case when all probabilities p_1, \dots, p_k are equal, and to the command in_{j_s} , which corresponds to the case when $p_s = 1$ and $p_t = 0$ for all $t \neq s$.

Another interesting possibility is to consider the three “events” *here*, *out*, *in* as having associated probabilities, that is, to move an object from a region to another one depending on the associated probability. Thus, each object will appear in the right hand member of rules $u \rightarrow v$ in the form $\alpha_{(here, p_1), (out, p_2), (in, p_3)}$, and it will remain in the same region, will exit it, or will enter a lower region with the probabilities p_1, p_2, p_3 , respectively.

This is again a generalization of known variants: if one of p_1, p_2, p_3 is equal to 1 and the other two are equal to 0, then we have the usual target indications *here*, *out*, *in*; when $p_1 = 0$ and $p_2 = p_3 = 0.5$ (that is, the commands *out* and *in* are equally probable) we cover the case of P systems with *immediate communication*: each object introduced by a rule has to exit the compartment where it is produced.

Because the probabilistic variants of P systems are generalizations of existing (non-probabilistic) variants, the families of sets of numbers or of languages generated in this way contain at least the families generated by usual systems. Whether or not the power is strictly increased and whether or not the probabilistic systems have other interesting/useful properties remains as a research topic.

6 Concluding Remarks

We have systematically considered the possibility to assign probabilities to several ingredients of P systems: to objects (and to their multiplicities), to evolution rules, to the target indications associated with the objects produced by rules. The approach is a preliminary one, an invitation to a closer exploration of these possibilities, either using mathematical tools or in the framework of a computer simulation (perhaps, dealing with a realistic case). Of course, also combinations of the above discussed possibilities can be considered; for instance, objects with probabilistic estimations of their multiplicities can be processed by rules having associated probabilities which determine their application. Expectedly, such combinations could be more adequate to real-life situations, but, at the same time, harder to handle.

Besides probabilistic approaches – and somewhat related in style – one can look for fuzzy set theory approaches (for instance, the probability that a copy of an object is present in a given membrane can be replaced by a subjective estimation, a membership degree in the fuzzy sense, see, e.g., [Negoiță, Ralescu, 1976], [Zadeh, 1965] – without requesting the restriction to have the sum of all probabilities associated with the same copy of an object equal to one). In the same line, of non-crisp mathematics, the problem remains open to consider rough set approaches, in the sense of [Pawlak, 1992].

References

- [Alberts et al., 1994] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.D., 1994. *Molecular Biology of the Cell*. 3rd ed., Garland Publishing, New York.
- [Ardelean, 2002] Ardelean, I.I., 2002. The Relevance of Biomembranes for P Systems. *Fundamenta Informaticae*, 49, 35–43.
- [Bernardini, Manca, 2002] Bernardini, F., Manca, V., 2002. Dynamical Aspects of P Systems. *BioSystems*, the present volume.
- [Bower, Bolouri, Eds., 2001] Bower, J.M., Bolouri, H., Eds., 2001. *Computational Modeling of Genetic and Biochemical Networks*. The MIT Press, Cambridge, Mass.
- [Dassow, Păun, 1989] Dassow, J., Păun, Gh., 1989. *Regulated Rewriting in Formal Language Theory*. Springer, Berlin.
- [Freund, 2002] Freund, R., 2002. Energy-controlled P systems. *Proc. of Workshop on Membrane Computing*, Curtea de Argeş, Romania, 2002, *LNCS*, Springer, to appear.
- [Frisco, Ji, 2002] Frisco, P., Ji, S., 2002. Info-energy P systems. *Proc. Eight Intern. Meeting on DNA Based Computers*, Sapporo 2002 (M. Hagiya, A. Obuchi, eds.), 161–170.
- [Koblitz, 1998] Koblitz, N., 1998. *Algebraic Aspects of Cryptography*. Springer, Berlin.
- [Madhu, 2002] Madhu, M., 2002. Probabilistic Rewriting P Systems. *Intern. J. Found. Computer Sci.*, to appear.
- [Marsan, 1990] Marsan, M.A., 1989. Stochastic Petri Nets. An Elementary Introduction. In *Advances in Petri Nets 1989* (G. Rozenberg, ed.), *LNCS* 424, Springer, Berlin, 1–29.
- [Moore, Crutchfield, 1997] Moore, C., Crutchfield, J.P., 1997. Quantum Automata and Quantum Grammars. *Santa Fe Institute Publ.*, 97-07-062.
- [Negoiță, Ralescu, 1976] Negoiță, C.V., Ralescu, D.A., 1976. *Applications of Fuzzy Sets Theory to Systems Analysis*. Birkhauser, Basel.
- [Nishida, 2002] Nishida, T.Y., 2002. Simulations of Photosynthesis by a K-Subset Transforming System with Membranes. *Fundamenta Informaticae*, 49, 1–3, 249–259.
- [Obtulowicz, 2002] Obtulowicz, A., 2002. Probabilistic P Systems. *Proc. of Workshop on Membrane Computing*, Curtea de Argeş, Romania, 2002, *LNCS*, Springer, to appear.
- [Păun, 2000] Păun, Gh., 2000. Computing with Membranes. *Journal of Computer and System Sciences*, 61, 1, 108–143, and Turku Center for Computer Science-TUCS Report No. 208, 1998 (www.tucs.fi).
- [Păun, 2001] Păun, Gh., 2001. From Cells to Computers: Computing with Membranes (P Systems). *BioSystems*, 59, 139–158.
- [Păun, 2002] Păun, Gh., 2002. *Membrane Computing. An Introduction*. Springer, Berlin.
- [Păun, Rozenberg, 2002] Păun, Gh., Rozenberg, G., 2002. A Guide to Membrane Computing. *Theoretical Computer Science*, 287, 73–100.
- [Pawlak, 1992] Pawlak, Z., 1992. *Rough Sets. Theoretical Aspects of Reasoning about Data*. Kluwer, Dordrecht.
- [Salomaa, 1969] Salomaa, A., 1969. Probabilistic and Weighted Grammars. *Information and Control*, 15, 529–544.
- [Suzuki et al., 2001] Suzuki, Y., Fujiwara, Y., Tanaka, H., Takabayashi, J., 2001. Artificial Life Applications of a Class of P Systems: Abstract Rewriting Systems on Multisets. In *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View* (C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), *LNCS*, 2235, Springer, Berlin, 299–346.
- [Zadeh, 1965] Zadeh, L., 1965. Fuzzy sets. *Information and Control*, 8, 338–353.

Appendix: Universality for a Class of Probabilistic P Systems

First, we define in a more formal way the class of systems we deal with, then we present some formal language theory prerequisites (matrix grammars with appearance checking, in the binary normal form), and after that we give the proof of the universality result.

A *rule-probabilistic rewriting P system* (of degree $m \geq 1$) is a construct

$$\Pi = (V, T, \mu, M_1, \dots, M_m, (R_1, \delta_1, \varphi_1), \dots, (R_m, \delta_m, \varphi_m)),$$

where V is the total alphabet, $T \subseteq V$ is the terminal alphabet, μ is a membrane structure (with m membranes, injectively labeled with $1, 2, \dots, m$), M_1, \dots, M_m are finite languages over V , representing the strings initially present in the m regions of μ , and $(R_i, \delta_i, \varphi_i)$, $1 \leq i \leq m$, are the rewriting rules present in region i of μ , together with the associated probabilities; the rules are of the form $X \rightarrow z(\text{tar})$, with $X \in V, z \in V^*$ and $\text{tar} \in \{\text{here}, \text{out}, \text{in}\}$, $\delta_i : R_i \rightarrow [0, 1]$ is the probability distribution to use a rule from R_i in the first step of a computation in Π , while $\varphi_i : R_i \rightarrow [0, 1]^{k_i}$, for $k_i = \text{card}(R_i)$ (we assume $R_i = \{r_{i,1}, \dots, r_{i,k_i}\}$), gives the probability distribution to use a rule $r_{i,t} \in R_i$ after using a rule $r_{i,s} \in R_i$; this probability is given in the t -th position of the vector $\varphi_i(r_{i,s})$. We assume that $\sum_{j=1}^{k_i} \delta_i(r_{i,j}) = 1$ for all $1 \leq i \leq m$, and that the sum of all components of each vector $\varphi_i(r_{i,j})$ is equal to 1, for all $1 \leq i \leq m$ and $1 \leq j \leq k_i$.

In what follows, we denote by $p(r_{i,s}, r_{i,t})$ the t -th component of the vector $\varphi_i(r_{i,s})$ (hence the probability to apply $r_{i,t}$ after $r_{i,s}$).

We start from the initial configuration of the system, that given by μ, M_1, \dots, M_m , and we proceed as usual in rewriting P systems, in each region rewriting each string which can be rewritten by a rule from that region. Each string is rewritten by at most one rule, and a string which cannot be rewritten remains unchanged. If a string $w = w_1 X w_2$ is rewritten by a rule $X \rightarrow z(\text{tar})$ in a region i , then the string $w' = w_1 z w_2$ is placed in the region indicated by tar : if $\text{tar} = \text{here}$, then the string w' remains in the region i , if $\text{tar} = \text{out}$, then the string is sent outside of the membrane i (in this way, strings rewritten in the skin region are sent out of the system), and if $\text{tar} = \text{in}$, then the string w' is sent to one of the membranes placed directly inside membrane i (if no such a membrane exists, then the rule cannot be applied).

What is new here is the way the probabilities are used to control the application of rules. At any moment, only rules with a non-zero probability can be used (in the end we accept a string only if the probability of its computation is strictly positive), and among these rules, the one which is applicable and has the maximal probability among the applicable rules is used (if several applicable rules have the same maximal probability, then one of them is non-deterministically chosen). Note that these probabilities depend, in each region separately and for each string separately, on the previous rule used for rewriting that string. If a string cannot be rewritten, then it is “lost”, because no rule can be ever used for rewriting it. In the initial step of a computation, in each region and for each string we use that applicable rule which has the maximal probability, as indicated by mappings $\delta_i, 1 \leq i \leq m$.

In the proof which follows we always have only one string in the system, and this simplifies the way of choosing the rule to be applied.

By using the rules as mentioned above, we get transitions among the configurations of the system, computations, and halting computations. As usual, we consider successful only halting computations. The result of a halting computation is the set of strings over T sent out of the system during the computation. The strings which remain inside the system as well as the strings sent out but containing symbols from $V - T$ are ignored; a computation which never ends gives no output.

A system Π with $T = V$ is said to be *non-extended*, hence the general case is referred to as *extended*.

We denote by $L_0(\Pi)$ the language of all strings computed as above by the system Π . Note that we have implicitly assumed that the probability of a computation is the product of the probabilities of all rules used during the computation and we have considered as generated by Π all terminal strings sent out of the system during halting computations which have a

strictly positive probability (we use $\lambda = 0$ as accepting threshold). That is why we denote by $ELSpP_m(0)$ the family of languages generated by extended probabilistic P systems with at most m membranes, $m \geq 1$; if non-extended systems are used, then the letter E is omitted.

In the universality proof below we need the notion of a *matrix grammar with appearance checking*, hence we briefly introduce it here. For further details, we refer to [Dassow, Păun, 1989].

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

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \rightarrow x_1, \dots, A_n \rightarrow x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either (1) $w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or (2) $w_i = w_{i+1}$, A_i does not appear in w_i , and the rule $A_i \rightarrow x_i$ appears in F . (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied – therefore we say that these rules are applied in the *appearance checking* mode.)

The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac} . It is known that $MAT_{ac} = RE$.

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in M are in one of the following forms:

1. $(S \rightarrow XA)$, with $X \in N_1, A \in N_2$,
2. $(X \rightarrow Y, A \rightarrow x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
3. $(X \rightarrow Y, A \rightarrow \#)$, with $X, Y \in N_1, A \in N_2$,
4. $(X \rightarrow \lambda, A \rightarrow x)$, with $X \in N_1, A \in N_2$, and $x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \rightarrow \#$ appearing in matrices of type 3; $\#$ is called a trap-symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of a derivation.

According to [Dassow, Păun, 1989], for each matrix grammar there is an equivalent matrix grammar in the binary normal form.

We are now ready to prove the universality result.

Theorem 6.1 $LSpP_1(0) = ELSpP_1(0) = RE$.

Proof. The inclusions \subseteq are obvious, hence it is sufficient to prove that $RE \subseteq LSpP_1(0)$. To this aim, we use the equality $RE = MAT_{ac}$ mentioned above. Let $G = (N, T, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, \#\}$. Assume that $N_2 = \{A_1, \dots, A_m\}$. Also, assume that M contains $n-1$ matrices of the types 2, 3, 4 (two-rule matrices): $m_i : (X \rightarrow Y, A \rightarrow x)$ matrices of type 2, for $1 \leq i \leq k_1$, $m_i : (X \rightarrow Y, A \rightarrow \#)$ matrices of type 3, for $k_1 + 1 \leq i \leq k_1 + k_2$, and $m_i : (X \rightarrow \lambda, A \rightarrow x)$ matrices of type 4, for $k_1 + k_2 + 1 \leq i \leq n-1$. The rules of each matrix m_i as above are labeled by $r_{i,1}, r_{i,2}, 1 \leq i \leq n-1$.

We construct the non-extended rule-probabilistic P system (of degree 1; the target indication here is omitted)

$$\Pi = (V, V, [1]_1, XAZH, (R_1, \delta_1, \varphi_1)),$$

where $(S \rightarrow XA)$ is the initial matrix of G and Z, H are new symbols, with

$$V = N_1 \cup N_2 \cup \{\#, Z, H\},$$

and the set R_1 and the probabilities are defined as follows.

1. For each matrix $m_i : (X \rightarrow Y, A \rightarrow x)$ with $1 \leq i \leq k_1$, we introduce the rules
 - $r_{i,1} : X \rightarrow Y$,
 - $r_{i,2} : A \rightarrow x$, with
 - $p(r_{i,1}, r_{i,2}) = 2/3$,
 - $p(r_{i,1}, r_n) = 1/3$ (the rule r_n will be specified below),
 - $p(r_{i,2}, r_{j,1}) = 1/n$, for all $1 \leq j \leq n-1$, and
 - $p(r_{i,2}, r_n) = 1/n$.
2. For each matrix $m_i : (X \rightarrow Y, A \rightarrow \#)$ with $k_1 + 1 \leq i \leq k_1 + k_2$, we introduce the rules
 - $r_{i,1} : X \rightarrow Y$,
 - $r_{i,2} : A \rightarrow \#$, with
 - $p(r_{i,1}, r_{i,2}) = 2/3$,
 - $p(r_{i,1}, r_{j,1}) = 1/(3n)$, for all $1 \leq j \leq n-1$,
 - $p(r_{i,1}, r_n) = 1/(3n)$, and
 - $p(r_{i,2}, r_n) = 1$.
3. For each matrix $m_i : (X \rightarrow \lambda, A \rightarrow x)$ with $k_1 + k_2 + 1 \leq i \leq n-1$, we introduce the rules
 - $r_{i,1} : X \rightarrow \lambda$,
 - $r_{i,2} : A \rightarrow x$.
 We also consider the following rules:
 - $r_Z : Z \rightarrow Z$,
 - $r'_Z : Z \rightarrow \lambda$,
 - $r_H : H \rightarrow \lambda(out)$, and
 - $r_{A_j} : A_j \rightarrow A_j$, for all $1 \leq j \leq m$.
 The probabilities are as follows:
 - $p(r_{i,1}, r_{i,2}) = 2/3$,
 - $p(r_{i,1}, r_n) = 1/3$,
 - $p(r_{i,2}, r_Z) = 1$,
 - $p(r_Z, r_{A_j}) = 2/(2m+1)$, for each $1 \leq j \leq m$, and
 - $p(r_Z, r'_Z) = 1/(2m+1)$,
 - $p(r'_Z, r_H) = 1$,
 - $p(r_H, r_H) = 1$,
 - $p(r_{A_j}, r_n) = 1$, for all $1 \leq j \leq m$.
4. Finally, we also consider the rule
 - $r_n : H \rightarrow H$, with
 - $p(r_n, r_n) = 1$.
 For all other pairs of rules (r, r') we have $p(r, r') = 0$.
5. The initial probabilities are
 - $\delta_1(r_{i,1}) = 1/n$, for all $1 \leq i \leq n-1$,
 - $\delta_1(r_n) = 1/n$,
 - and for all other rules r we have $\delta_1(r) = 0$.

Note that for all rules r the vector $\varphi_1(r)$ gives a probability distribution over R_1 .

The work of this system is easily seen in the representation from Figure 4 (clearly, the dots and the black boxes at the end of certain arrows indicate that those arrows go to the arrows starting from a dot or a box, respectively).

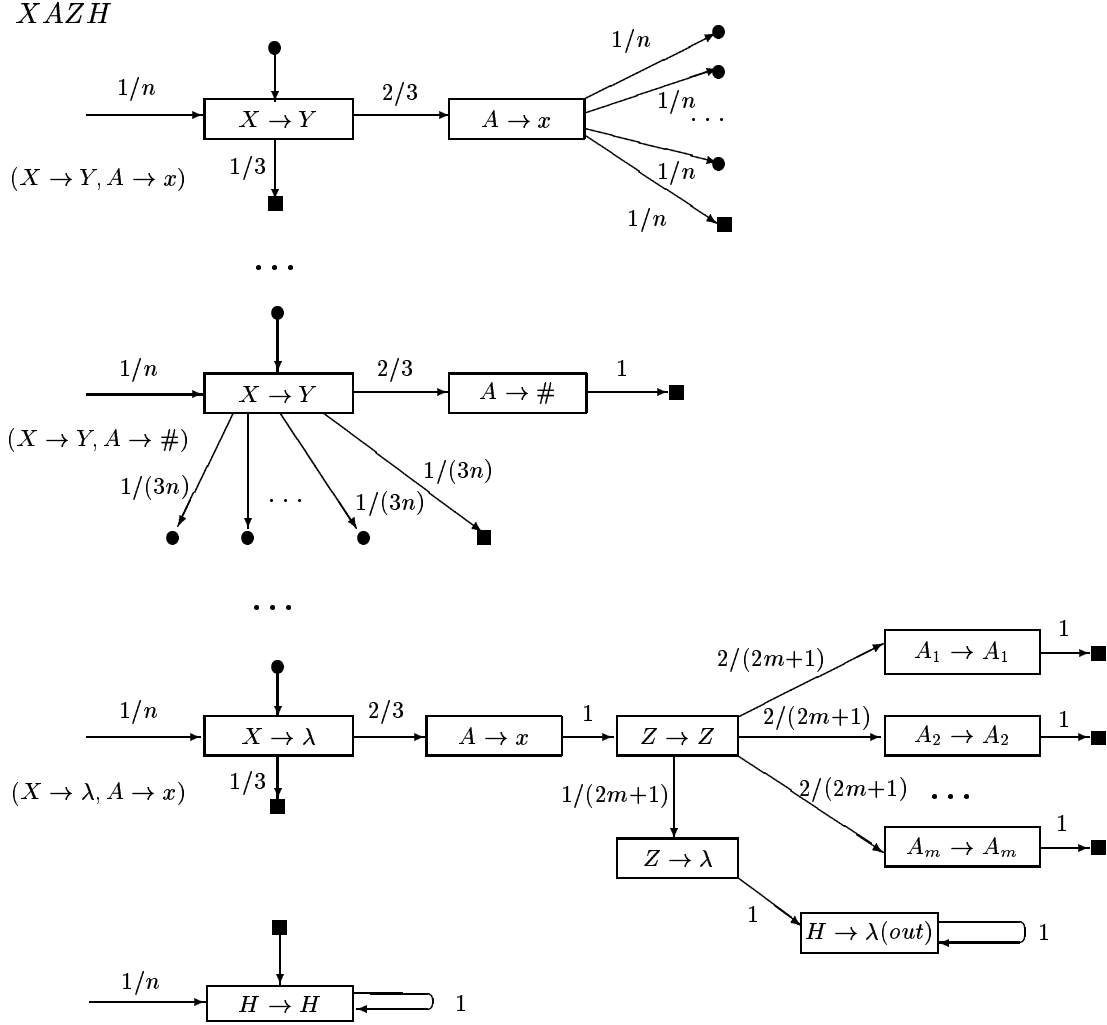


Figure 4: The P system from the universality proof

We start with the string $XAZH$, where XA are the symbols introduced by the initial matrix of G and Z, H are symbols not in $N \cup T$. If the rule $r_n : H \rightarrow H$ is ever used, then the computation will never stop, hence no output is obtained.

At the beginning, any rule $r_{i,1}, 1 \leq i \leq n-1$, can be used (they have equal probabilities as given by δ_1), hence we start the simulation of any matrix of G . (Also r_n can be used, but this will lead to no output.) In all cases, the next rule to be applied is, with probability $2/3$, the second rule of the same matrix, $r_{i,2}$.

If we have a matrix of type 2 and $r_{i,2}$ can be used, then we can continue with any rule $r_{j,1}, 1 \leq j \leq n-1$, hence passing to simulating another matrix, or, if no such rule is applicable, with r_n , hence the computation will never end. If the second rule of the matrix $m_i, 1 \leq i \leq k_1$,

cannot be used, then we have to use r_n (it has probability $1/3$, and can always be applied).

If we have a matrix of type 3 and $r_{i,2}$ can be used, then we continue with r_n and no output is obtained. If the rule $r_{i,2} : A \rightarrow \#$ cannot be used (this means that A is not present in the current string), then we are allowed to choose any rule $r_{j,1}$, $1 \leq j \leq n-1$, or r_n ; the latter case is obligatory when the derivation in G cannot continue.

Therefore, each matrix of types 2, 3 is correctly simulated in Π if and only if the rule r_n is not used. In this way, each (terminal) derivation in G can be simulated in Π . When using $r_{i,1}$ for a matrix of type 4, we again go to $r_{i,2}$ with probability $2/3$ and to r_n with probability $1/3$, which ensures the correct simulation of the matrix.

However, after using a matrix m_i of type 4 we have to check whether or not the obtained string is terminal, and only in the affirmative case we have to send it outside the system and stop the computation. This is done as follows. After using the rule $r_{i,2}$ we use the rule $r_Z : Z \rightarrow \lambda$. All rules r_{A_j} , $1 \leq j \leq m$, have a greater probability than the rule $r'_Z : Z \rightarrow \lambda$, hence if any symbol A_j is still present in the string, then the corresponding rule $A_j \rightarrow A_j$ is used, and it is followed by r_n , hence the computation never halts. Only if the string contains no symbol from N_2 the rule r'_Z can be used, and the symbol Z is erased. We know in this way that the derivation in G was a terminal one. We also erase the symbol H and we send the obtained string out of the system.

Consequently, $L(G) = L(\Pi)$, and this completes the proof. \square

Note that the previous proof can be slightly simplified if we want to prove only the equality $ELSpP_1(0) = RE$, because in the extended case we do not need to check whether or not the obtained string is terminal with respect to G : we take T as the terminal alphabet of Π , too, and we replace $p(r_{i,2}, r_Z) = 1$ with $p(r_{i,2}, r_H) = 1$ for $k_1 + k_2 + 1 \leq i \leq n-1$, avoiding the use of the symbol Z and of rules r_Z, r'_Z, r_{A_j} , $1 \leq j \leq m$.