# Self-constructing Recognizer P Systems

Daniel Díaz-Pernil[1], Francisco Peña-Cantillana[2],
Miguel A. Gutiérrez-Naranjo[2]

[1]Research Group on Computational Topology and Applied Mathematics
Department of Applied Mathematics - University of Sevilla, 41012, Spain
sbdani@us.es

[2]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, 41012, Spain
frapencan@gmail.com, magutier@us.es

**Summary.** Usually, the changes produced in the membrane structure of a P system are considered side effects. The output of the computation is encoded as a multiset placed in a specific region and the membrane structure in the halting configuration is not considered important. In this paper we explore P systems where the target of the computation is the construction of a new membrane structure according its set of rules. The new membrane structure can be considered as the initial one of a new *self-constructed* P system. We focus on the self-construction of recognizer P systems and illustrates the definition with a study of the self-construction P systems working as decision trees for solving Machine Learning decision problems.

## 1 Introduction

In many Membrane Computing models, changing the membrane structure of a P system along the computation is a common process. The changes are produced via division of membranes (based on *cellular mitosis*), via creation of new membranes from objects (based on *cellular autopoiesis*, see [7]) or dissolution of membranes. The rules producing such changes have been deeply studied and the capability of the P systems for solving hard problems are linked to the use of such rules (see, e.g., [4, 5, 16]).

Nonetheless, the changes of the membrane structure produced along a computation are not considered a target itself. The changes are usually produced in order to compute an *output*, which is usually encoded as a multiset (or as a single distinguished object) in the corresponding output region. The membrane structure obtained in the halting configuration is not important. It is merely a collateral effect.

In this paper, we focus on P systems where the target of the computation is exactly the opposite to the usual one. We study P systems whose aim is to

develop a membrane structure. Such P systems will take a multiset as *input* and they will change their membrane structure according to such input, the set of rules and the non-deterministic choices, if any. The membrane structure obtained in the halting configuration will be considered as the *output* of the computation. This new membrane structure, together the remaining ingredients of the P system (alphabet, set of labels, set of rules, ...) can be considered as a new P system, able to receive a new *input* and perform a new computation. In this way, we will consider that this *second* P system (which is similar to the original one, but with a new initial membrane structure) has been self-constructed, since a (potentially) complex membrane structure has been obtained from a simple one (maybe from an initial membrane structure with the skin as unique membrane) according to the application of their own rules. Of course, different final membrane structures may be obtained from different inputs, but also with the same input due to the non-determinism.

The self-construction of a complex membrane structure can be a target by itself, as shown in [3, 6], but in this paper, the self-constructed P system is thought for a second use. From this general target, we focus here on the self-construction of *recognizer P systems*, i.e., the P system with this new membrane structure can be now used as recognizer P systems for solving decision problems in the usual way: An instance of the decision problem is provided to the P system as an input encoded as an appropriate multiset and an object *yes* or *no* (but no both) is sent to the environment in the last step of the computation.

In this way, two different uses for the P system are considered:

- Firstly, given a P system, a multiset is placed in the corresponding input membrane and the computation starts. According to the input and the non deterministic choice of applicable rules, the initial membrane structure is modified along the computation. As usual, a halting configuration is reached if no more rules can be applied. The self-construction of the recognizer P system is finished.
- Secondly, we consider a new computation of the P system, but in this stage, the membrane structure obtained in the halting configuration of the previous stage is considered as the initial one. This new computation also needs a new input, which is placed in the corresponding input membrane. In this stage, the *output* will be a specific object (*yes* or *no*, but no both) which is placed in the output region in the halting configuration.

The paper is organized as follows: Next, we recall the definition of recognizer P system used in this paper. In Section 3, our case study is presented, the self-construction of a P system from a training set which works as a decision tree and the classification (decision problem) of new instances as in Machine Learning theory. We provide the formal framework, the general construction of the P systems, an example and some theoretical considerations. Finally, Section 4 finishes the paper with some conclusions.

## 2 Recognizer P Systems

Recognizer P systems were introduced in [11] and they are the natural framework to study and solve decision problems, i.e., problems were a Boolean total function $\theta_X$ must be defined on a set of instances $I_X$. Recognizer P systems are associated in a natural way with P systems with *input* and with external *output*, i.e., each instance of the problem is codified by a multiset placed in an *input membrane*. The output of the computation (*yes* or *no*) is sent to the environment. Due to the non-determinism, the definition of recognizer P system claims that the output of *all* the computations must be the same. Since one can find slightly different approaches in the literature (see [9, 15]), we recall the definition used in this paper:

**Definition 1.** *A* P system with input *is a tuple* $(\Pi, \Sigma, i_\Pi)$*, where: (a)* $\Pi$ *is a P system, with working alphabet* $\Gamma$*, with* $p$ *membranes labelled by* $1, \ldots, p$*, and initial multisets* $w_1, \ldots, w_p$ *associated with them; (b)* $\Sigma$ *is an (input) alphabet strictly contained in* $\Gamma$*; the initial multisets are over* $\Gamma - \Sigma$*; and (c)* $i_\Pi$ *is the label of a distinguished (input) membrane.*

Let $m$ be a multiset over $\Sigma$. The *initial configuration of* $(\Pi, \Sigma, i_\Pi)$ *with input* $m$ is $(\mu, w_1, \ldots, w_{i_\Pi} \cup m, \ldots w_p)$.

**Definition 2.** *A* recognizer P system *is a P system with input,* $(\Pi, \Sigma, i_\Pi)$*, and with external output such that:*

1. *The working alphabet contains two distinguished elements yes, no.*
2. *All its computations halt.*
3. *If* $\mathcal{C}$ *is a computation of* $\Pi$*, then either some object yes or some object no (but not both) must have been released into the environment, and only in the last step of the computation. We say that* $\mathcal{C}$ *is an accepting computation (respectively, rejecting computation) if the object yes (respectively, no) appears in the external environment associated to the corresponding halting configuration of* $\mathcal{C}$*.*

## 3 A Case Study: Decision Trees

Decision trees is one of the most widely used structures in Computer Science. They are used to classify an input by sorting it down the tree from the root to some leaf node, which provides the classification of the instance. Instances are usually written as sets of pairs $\langle Attribute, Value \rangle$ and each node in the tree determines a test of some attribute. Each branch descending from that node corresponds to one of the possible value for this attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified in this node and moving down the branch corresponding to the value of the instance in this attribute. The leaves are labelled with values of the classification and they are the output associated to the instances that reach them. In this way, if the possible classifications are

**Table 1.** A classic example of training set adapted from [14].

| Day | *Outlook* | *Temperature* | *Humidity* | *Wind* | *PlayTennis* |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

$YES$ and $NO$, a decision tree can be thought as a Boolean mapping on the set of instances which *decides* the classification of the instance according to the values of the attributes.

Let us illustrate the process with a classic example adapted from [14]. It consists on a database with fourteen days. Each day is represented by the values of the attributes *Outlook*, *Temperature*, *Humidity* and *Wind*. Each day has also associated its classification with respect to the attribute *PlayTennis* (see Table 1). From a learning point of view, the database can be seen as a training set. The target is to generate a decision tree from this database which can be used to classify *new* instances.

Figure 1 shows a decision tree consistent with the training set shown in Table 1, i.e., a tree which classifies correctly all the examples in the training set. According to this tree, a day with $Outlook = Sunny$, $Temperature = Cool$, $Humidity = Normal$ and $Wind = Strong$ (which does not belong to the training set) will be classified as $YES$.

### 3.1 The P System Model

The definition of self-construction in P system is independent of the P system model, i.e., it can be considered in the framework of cell-like, tissue-like or whatever other graph structure and it can be adapted to different semantics. The unique restriction that the P system must satisfy is the ability of modifying the initial membrane structure according to the input. In this way, the concept can be considered in many scenarios.

In this paper, we will illustrate the definition with a P system model were the data are encoded as strings [1, 13] and the changes in the membrane structure are performed via membrane creation [5, 10].

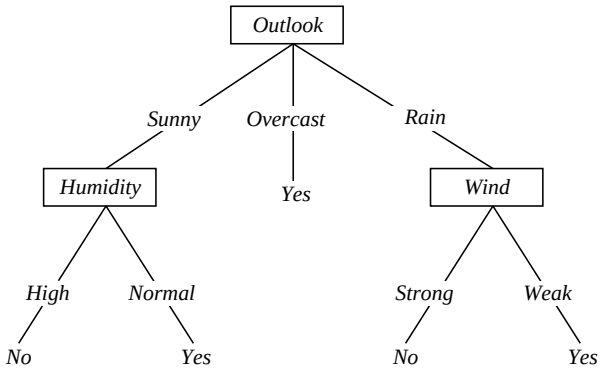


**Fig. 1.** An example of tree obtained from the training set shown in 1. The image is available from `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html`.

Formally, a *self-constructing P system with strings and membrane creation* is a construct of the form $\Pi = (O, H, L, R)$ where:

1. $O$ is the alphabet of *objects*;
2. $H$ is a finite set of labels;
3. $L$ is a finite languages over $O$;
4. $R$ is a finite set of *rules*, of the following forms:
   a) $[w_p + w_q \rightarrow \sum_k w_k]_h$ where $h \in H$, $w_p$, $w_q$ and $w_k$ are strings over $O$. These are *2-cooperative evolution rules*: The simultaneous occurrence of the strings $w_p$ and $w_q$ in the membrane $h$ produces a finite set of strings in the same membrane. As usual, $w_p$ and $w_q$ are consumed.
   b) $w_p[\,]_h \rightarrow [w_q]_h$ where $h \in H$; $w_p$ and $w_q$ are strings over $O$. These are *send-in communication rules*. A string is introduced in the membrane possibly modified.
   c) $[w_p]_h \rightarrow [\,]_h\, w_q$ where $h \in H$; $w_p$ and $w_q$ are strings over $O$. These are *send-out communication rules*. A string is sent out of the membrane possibly modified.
   d) $[a \rightarrow [M]_{h_2}]_{h_1}$ where $h_1, h_2 \in H$, $a \in O$ and $M$ is a finite language over $O$. These are *creation rules*. An object $a$ placed in a membrane with label $h_1$ creates a new membrane with label $h_2$. This new membrane has associated an initial finite language $M$.

We will consider that the *self-constructing P system with strings and membrane creation* always has a unique membrane (the skin) in the initial membrane structure, such membrane is the input membrane and the output region is the environment. The multiset $L$ is placed in the skin at the initial configuration. Rules are applied according to the following principles:

- Rules are used as usual in the framework of Membrane Computing, that is, in a maximal parallel way. In one step, each string in a membrane can only be used

for one rule (non deterministically chosen when there are several possibilities), but any string which can evolve by a rule of any form must do it (with the restrictions below indicated).
- All the strings which are not involved in any of the operations to be applied remain unchanged.
- The rules associated with the label $h$ are used for all membranes with this label, irrespective of whether or not the membrane is an initial one or it was obtained by creation.
- Several rules can be applied to different objects in the same membrane simultaneously.

### 3.2 Self-constructing P Systems for Decision Trees Learning

Next, we will provide a family of *self-constructing P system with strings and membrane creation* for Decision Trees Learning. Such self-constructed P system will receive instances of the problem and will output *yes* or *no*, i.e., they are tools for solving decision problems.

In a first stage, the P system takes a finite language, codifying a *training set*, as input. Each example in the training set will be encoded as a string. The set of these strings will be the initial language $L$ and it will be placed in the unique initial membrane of the P system, as defined above. After a finite number of steps, the computation halts and the membrane structure has been (probably) modified. The P system with this halting membrane structure is the recognizer P system which has been self-constructed according with the training set provided as input. The self-constructed recognizer P system with the membrane structure obtained in the halting configuration is now prepared to receive an *instance* of the decision problem and will provide an answer *yes* or *not*.

Let us start by considering a training set $D = \{(v_1, c_1), \ldots, (v_n, c_n)\}$ where, for each $i \in \{1, \ldots, n\}$, $v_i$ is a tuple of pairs $\langle Attribute, Value\rangle$ and $c_i = \{YES, NO\}$ is the classification for a concept[1]. We will consider a set of attributes $ATR = \{A_1, \ldots, A_k\}$ and, for each $i \in \{1, \ldots, k\}$, $VAL_i = \{v_i^1, \ldots, v_i^{j_i}\}$ is the set of values of the attribute $A_i$. We will consider that the sets $VAL_i$ are disjoint pairwise. We will also consider $VAL = VAL_1 \cup \ldots \cup VAL_k$ and $\Gamma = ATR \cup VAL \cup \{YES, NO\}$. The set $\Gamma$ will be called a *training set alphabet*. Notice that many different training sets can have the same alphabet $\Gamma$.

By using this notation, we will represent each example $(v_i, c_i)$ as a string over the set $\Gamma$ and a training set can be considered as a finite language over this set. The example $(v_i, c_i)$ will be represented by the string of $2k + 1$ symbols $A_1 v_1^i A_2 v_2^i \ldots A_k v_k^i c_i$, where $v_j^i$ is the value of the attribute $A_j$ in the $i - th$ example of the training set and $c_i \in \{YES, NO\}$ is the value of the classification. For example, the string

[1] A formal description of the principles of Machine Learning is out of the scope of this paper. A detailed introduction can be found in, e.g., [8].

$$Outlook\, Sunny\, Temperature\, Hot\, Humidity\, High\, Wind\, Weak\, NO$$

is the representation of the first example of Table 1.

In the first stage, the finite language codifying the training set is placed in the skin and the self-construction starts. When it finishes, the halting membrane structure is prepared for accepting new inputs and *deciding* on it. The new input will be similar to the encoding of an example, but the last object of the string will be ? instead of $YES$ or $NO$. For example, in order to know the classification of a day not with $Outlook = Sunny$, $Temperature = Cool$, $Humidity = Normal$ and $Wind = Strong$, the string

$$Outlook\, Sunny\, Temperature\, Cool\, Humidity\, Normal\, Wind\, Strong\, ?$$

will be placed in the input membrane (the skin) and a new computation will start.

Next we provide the formal definition of a *self-constructing P system with strings and membrane creation* associated to a training set $\Gamma$. The P system is a 4-uple $\Pi = (O, H, L, R)$ where:

1. $O = \Gamma \cup \{YES_{aux}, YES_{act}, ?, NO_{aux}, NO_{act}, new\}$ is the alphabet of objects;
2. $H = \{skin\} \cup VAL$. The possible labels are the values of the attributes plus the initial label $skin$;
3. $L = \{YES_{aux}, NO_{aux}\}$ Two strings, each of them with only one object, are placed in the skin in the initial configuration;
4. We split the set $R$ of rules into two groups, the rules used in the self-construction stage and the rules used in the decision stage:

**Rules for the self-construction stage.**

**R1**. $\left.\begin{array}{l}[\overline{x}YES + YES_{aux} \to \overline{x}YES + YES_{act}\,]_h \\ [\overline{x}NO + NO_{aux} \to \overline{x}NO + NO_{act}\,]_h\end{array}\right\}$ for $h \in H$.

where $\overline{x}$ is a string over $\Gamma$ composed by pairs $(Attribute, Value)$. If the string $\overline{x}YES$ and $YES_{aux}$ occur simultaneously in the same membrane, then $\overline{x}YES$ remains unchanged, but $YES_{aux}$ is consumed and $YES_{act}$ is produced. Analogously for the $NO$ case.

**R2**. $[YES_{act} + NO_{act} \to new]_h$ for $h \in H$.

If the strings $YES_{act}$ and $NO_{act}$ are placed simultaneously in the same membrane, both are consumed and the string $new$ is produced.

**R3**. $[new + \overline{x}A_i\overline{y} \to \overline{x}A_i\overline{y} + v_1 + \ldots + v_s]_h$ for $h \in H$.

If the strings $new$ and $\overline{x}A_i\overline{y}$ occur in the same membrane (where $\overline{x}A_i\overline{y}$ is a string including the object $A_i$, which denotes an attribute), then the string $new$ is consumed, the string $\overline{x}A_i\overline{y}$ remains unchanged and all the strings $v_j^i$ from $VAL_i$ are produced. Let us notice that this set of rules produces a high

degree of non-determinism. On the one hand, many different strings $\overline{x}A_i\overline{y}$ can simultaneously occur in the same membrane and, on the other hand, several $A_i$ may be chosen from the same string. Nonetheless, since there can exists at most only one string *new* in each membrane at each time unit, only one of these rules will be applied.

**R4**. $[v \to [\, YES_{aux}\, NO_{aux}]_v\,]_h$ for $h \in H$
Each string $v \in VAL$ creates a new membrane. Such membrane will have $v$ as a label and it will contain the strings $YES_{aux}$ and $NO_{aux}$.

**R5**. $\overline{x}A_iv\overline{y}\,[\,]_v \to [\,\overline{xy}\,]_v$ for $A_i \in ATR$ and $v \in VAL_i$
Each string $\overline{x}A_iv\overline{y}$ (where $A_i$ is an object which denotes an attribute and $v$ is the next symbol in the string, denoting one of the values of $A_i$) out of a membrane with label $v$ will be sent into the membrane. The application of the rule will transform the string into $\overline{xy}$, which is $\overline{x}A_iv\overline{y}$ after deleting the substring $A_iv$. These rules are applied in parallel and several strings can cross out the same membrane simultaneously.

**Rules for the decision stage.**

**R6**. $\overline{x}Av\overline{y}?\,[\,]_v \to [\,\overline{xy}?\,]_v$ for $v \in VAL$.
If a string ended with ? and containing an object $v \in VAL$ is out of a membrane with label $v$, then the string is sent into the membrane. The application of the rule also produces a change in the string since the object $v$ and the previous object in the string (the object $A$ denoting the corresponding attribute) are deleted.

**R7**. $\left.\begin{array}{l}[\overline{x}? + YES_{act} \to YES_{out} + YES_{act}\,]_h \\ [\overline{x}? + NO_{act} \to NO_{out} + NO_{act}\,]_h\end{array}\right\}$ for $h \in H$.
where $\overline{x}$ is a string over $\Gamma$ composed by pairs $(Attribute, Value)$. If the string $\overline{x}?$ and $YES_{act}$ occur simultaneously in the same membrane, then $YES_{act}$ remains unchanged, but $\overline{x}?$ is consumed and $YES_{out}$ is produced. Analogously for the $NO$ case.

**R8**. $\left.\begin{array}{l}[\,YES_{out}\,]_h \to [\,]_h\, YES_{out} \\ [\,NO_{out}\,]_h \to [\,]_h\, NO_{out}\end{array}\right\}$ for $h \in H$.
when an object $YES_{out}$ (resp. $NO_{out}$) is produced, the decision is made. This set of rules sends such object from the membrane where is produced to the environment. Such objects are the answers to the decision problem

$Outlook\, sunny\, Temperature\, hot\, Humidity\, high\, Wind\, weak\, NO$
$Outlook\, sunny\, Temperature\, hot\, Humidity\, high\, Wind\, strong\, NO$
$Outlook\, overcast\, Temperature\, hot\, Humidity\, high\, Wind\, weak\, YES$
$Outlook\, rain\, Temperature\, mild\, Humidity\, high\, Wind\, weak\, YES$
$Outlook\, rain\, Temperature\, cool\, Humidity\, normal\, Wind\, weak\, YES$
$Outlook\, rain\, Temperature\, cool\, Humidity\, normal\, Wind\, strong\, NO$
$Outlook\, overcast\, Temperature\, cool\, Humidity\, normal\, Wind\, strong\, YES$
$Outlook\, sunny\, Temperature\, mild\, Humidity\, high\, Wind\, weak\, NO$
$Outlook\, sunny\, Temperature\, cool\, Humidity\, normal\, Wind\, weak\, YES$
$Outlook\, rain\, Temperature\, mild\, Humidity\, normal\, Wind\, weak\, YES$
$Outlook\, sunny\, Temperature\, mild\, Humidity\, normal\, Wind\, strong\, YES$
$Outlook\, overcast\, Temperature\, mild\, Humidity\, high\, Wind\, strong\, YES$
$Outlook\, overcast\, Temperature\, hot\, Humidity\, normal\, Wind\, weak\, YES$
$Outlook\, rain\, Temperature\, mild\, Humidity\, high\, Wind\, strong\, NO$

**Fig. 2.** Finite language encoding the training set from Table 1.

### 3.3 An example

As an example of self-construction P systems for Decision Tree Learning, let us consider the training set from Table 1. According to the encoding previously described, such training set can be written as shown in Fig. 2.

Let us consider an initial configuration $C_0$ which has only one membrane with label *skin*. Such membrane contains the language codifying the training set from Fig. 2, together with $YES_{aux}$ and $NO_{aux}$. From this initial configuration only two rules from **R1** are applied. The application of such rules consumes $YES_{aux}$ and $NO_{aux}$ and produces $YES_{act}$ and $NO_{act}$. In the second step of the computation, only the rule from **R2** is applied. The objects $YES_{act}$ and $NO_{act}$ are consumed and *new* appears in the skin. In this way, the configuration $C_2$ has only one membrane, the *skin*, where the codification of the training set and the object *new* are placed.

From this configuration $C_2$, one and only one of the rules from **R3** is nondeterministically chosen and applied. In the choice, one of the strings encoding an example from the training set is taken and in this string, one of the objects encoding an attribute is also selected. Let us suppose that the string

$$Outlook\, sunny\, Temperature\, hot\, Humidity\, high\, Wind\, weak\, NO$$

is chosen and the object *Outlook* is selected. The application of the rule consumes the object *new*, keeps unchanged the string encoding the example and three new objects *rain*, *sunny* and *overcast* appear. Therefore, the configuration $C_3$ has only one membrane, the *skin*, where the codification of the training set and the objects *rain*, *sunny* and *overcast* are placed.

In the next step, the changes in the membrane structure start. The objects *rain*, *sunny* and *overcast* create new membranes. Each membrane has the objects $YES_{aux}$ and $NO_{aux}$ inside and the corresponding value *rain*, *sunny* or *overcast* as label, i.e.,

$$C_4 = \left[ \begin{array}{cc} TS & [\, YES_{aux}\, NO_{aux}\,]_{rain} \\ [\, YES_{aux}\, NO_{aux}\,]_{sunny} & [\, YES_{aux}\, NO_{aux}\,]_{overcast} \end{array} \right]_{skin}$$

where $TS$ represents the language encoding the training set. In the next step, rules from **R5** are applied. All the strings in the skin are sent into the new membranes with slight changes. These new strings in the elementary membranes are the following. The set $TR_{sunny}$ of strings in the membrane with label *sunny* is

$$\begin{array}{l} Temperature\, hot\, Humidity\, high\, Wind\, weak\, NO \\ Temperature\, hot\, Humidity\, high\, Wind\, strong\, NO \\ Temperature\, mild\, Humidity\, high\, Wind\, weak\, NO \\ Temperature\, cool\, Humidity\, normal\, Wind\, weak\, YES \\ Temperature\, mild\, Humidity\, normal\, Wind\, strong\, YES \end{array}$$

the set $TR_{rain}$ of strings in the membrane with label *rain* is

$$\begin{array}{l} Temperature\, mild\, Humidity\, high\, Wind\, weak\, YES \\ Temperature\, cool\, Humidity\, normal\, Wind\, weak\, YES \\ Temperature\, cool\, Humidity\, normal\, Wind\, strong\, NO \\ Temperature\, mild\, Humidity\, normal\, Wind\, weak\, YES \\ Temperature\, mild\, Humidity\, high\, Wind\, strong\, NO \end{array}$$

and, finally the set $TR_{overcast}$ of strings in the membrane with label *overcast* is

$$\begin{array}{l} Temperature\, hot\, Humidity\, high\, Wind\, weak\, YES \\ Temperature\, cool\, Humidity\, normal\, Wind\, strong\, YES \\ Temperature\, mild\, Humidity\, high\, Wind\, strong\, YES \\ Temperature\, hot\, Humidity\, normal\, Wind\, weak\, YES \end{array}$$

In the configuration $C_5$, the membrane with label *sunny* has five strings encoding examples and two objects $YES_{aux}$ and $NO_{aux}$. The situation is similar to the initial configuration, where the membrane *skin* had fourteen strings encoding examples. Let us consider that from the configuration $C_8$, the chosen rule from **R3** takes the string

$$Temperature\, hot\, Humidity\, high\, Wind\, weak\, NO$$

and the object $Humidity$. In $C_9$, two new membranes appear inside the membrane with label *sunny*, one of them with membrane *high* and the other one with label *normal*. In the configuration $C_{10}$, this new membrane with label *high* contains the objects $YES_{aux}$ and $NO_{aux}$ and the set of strings $TR_{sunny+high}$

$$\begin{array}{l} Temperature\, hot\, Wind\, weak\, NO \\ Temperature\, hot\, Wind\, strong\, NO \\ Temperature\, mild\, Wind\, weak\, NO \end{array}$$

analogously, the new membrane with label *normal* contains the objects $YES_{aux}$ and $NO_{aux}$ and the set of strings $TR_{sunny+normal}$

$$Temperature\, cool\, Wind\, weak\, YES$$
$$Temperature\, mild\, Wind\, strong\, YES$$

In a similar process, if the chosen rule from **R3** in the membrane with label *rain* selects an object $Wind$ from the taken string, then, such membrane will have two new membranes inside in the configuration $C_{10}$. One on them, with label *strong* will contain the objects $YES_{aux}$ and $NO_{aux}$ and the set of strings $TR_{rain+strong}$

$$Temperature\, cool\, Humidity\, normal\, NO$$
$$Temperature\, mild\, Humidity\, high\, NO$$

The second new membrane, with label *weak* will contain the objects $YES_{aux}$ and $NO_{aux}$ and the set of strings $TR_{rain+weak}$

$$Temperature\, mild\, Humidity\, high\, YES$$
$$Temperature\, cool\, Humidity\, normal\, YES$$
$$Temperature\, mild\, Humidity\, normal\, YES$$

Let us consider now the membrane with label *overcast* in the configuration $C_5$. It contains the objects $YES_{aux}$ and $NO_{aux}$ and the set of strings $TR_{overcast}$. In the next step, the object $YES_{aux}$ is transformed into $YES_{act}$ by application of one rule from **R1**, but $NO_{aux}$ keeps unchanged, since there is no string with $NO$ as the last object. This means that no more rules can be applied and the computation in this membrane finishes. The same reasoning is valid for the remaining membranes where all the strings end in $YES$ or $NO$. In this way, the configuration $C_{11}$ is a halting one and the decision P systems is already constructed (see Fig. 3).

This self-constructed recognizer P system can be used now for deciding on new instances. Let us consider a new computation. The target is to obtain a Boolean answer as a classification for the day with $Outlook = Sunny$, $Temperature = Cool$, $Humidity = Normal$ and $Wind = Strong$. In such way, the string

$$Outlook\, Sunny\, Temperature\, Cool\, Humidity\, Normal\, Wind\, Strong\, ?$$

will be placed in the input membrane (the skin) and this is the unique string in the skin in the new configuration $C_0$. The corresponding rule from **R6** is applied and the string is sent to the membrane with label *sunny* slightly modified:

$$Temperature\, Cool\, Humidity\, Normal\, Wind\, Strong\, ?$$

In the next step, a new rule from **R6** is applied and the string $Temperature\, Cool\, Wind\, Strong\, ?$ is placed in the membrane with label *normal* in the configuration $C_2$. Since $YES_{act}$ occurs in this membrane, the string is consumed and $YES_{out}$ appears in the configuration $C_3$. By three applications of rules from **R7**, the object $YES_{out}$ is sent to the environment and the configuration $C_6$ is a halting configuration. The answer $YES_{out}$ is sent to the environment in the last step of the computation and it is the answer corresponding to the input in this recognizer P system.

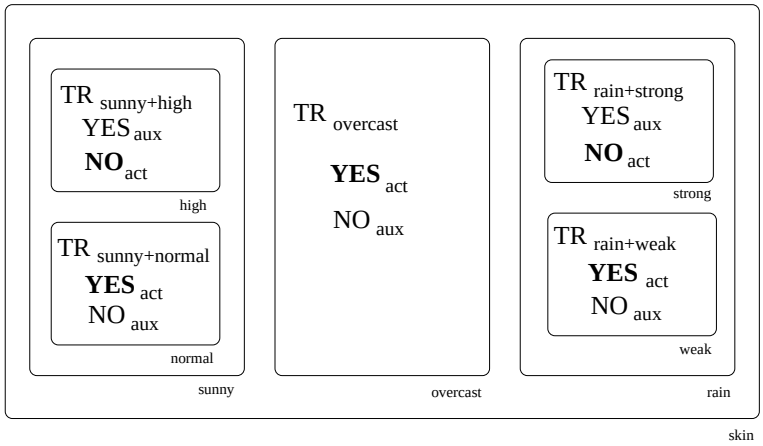


**Fig. 3.** Halting configuration for the self-constructed P system.

### 3.4 Theoretical Foundations

Next, we provide several considerations about the algorithm provided in the previous section. The first one is related to the training set provided as *input*. It must be carefully checked in order to avoid *noise*, i.e., it is not acceptable that the same instance appears with two different classifications in the same training set. Following the example from Table 1, we do not accept that two days with the same values for the attributes $Outlook$, $Temperature$, $Humidity$ and $Wind$ have different classifications for $PlayTennis$.

If the training set is free of noise, then the self-construction of the recognizer P system (first use) and the classification of new instances (second use) halt after a finite number of steps. A deeper question is the predictive power of the recognizer P system on new instances. A detailed study of such question is out of the scope of this paper. Nonetheless, let us briefly notice that usually, more than one decision tree is derivable from a training set and they may be not equivalent. From a Membrane Computing point of view, it is clear that the non-determinism produced by the set of rules **R3** produces a big amount of possible halting configurations.

As an illustrative example, let us consider a training set on wooden toy blocks with two attributes *Color* and *Shape* and the classification for *BelongsToEddy* shown in Table 2. Figure 4 shows two different trees consistent with the training set from Table 2. Both are *consistent*, since they classify correctly all the instances of the training set, but they are not equivalent. For example, the classification of a green and squared block, an instance which does not belongs to the training set, depends on the chosen tree. In Machine Learning, the followed criterion is *entia non sunt multiplicanda praeter necessitatem*, known as Ockham's razor[2] which states that among competing hypotheses, the one with the fewest assumptions

[2] Due to William of Ockham (1287 − 1347), see [8] for details.

**Table 2.** An example of training set for toy blocks.

| Block | *Color* | *Shape* | *BelongsToEddy* |
|---|---|---|---|
| B1 | Red | Square | Yes |
| B2 | Red | Triangle | Yes |
| B3 | Green | Triangle | No |
| B4 | Green | Circle | No |

should be selected. In this paper, we consider that all the trees consistent with the training set can be a solution of the problem and do not apply any *bias* for selecting a specific one.

The theoretical correctness, from a Machine Learning point of view, can be summed up in the following theorem.

**Theorem 1.** *Let us consider a non-empty training set $D$ without noise. Let us consider that the instances in $D$ has $k$ attributes. Let $\Pi$ be the self-constructing P system associated to $D$ as shown above.*

**(C1)** *In the self-constructing stage, in any possible computation, there exists $p \in \{0, \ldots, k\}$ such that $C_{1+5p}$ is a halting configuration. Therefore, each computation gives at most $1 + 5k$ steps.*

**(C2)** *Any computation in the decision stage gives $2p + 2$ steps with $p \in \{0, \ldots, k\}$ and sends to the environment $YES_{act}$ or $NO_{act}$ (but no both) in the last step of computation.*

**(C3)** *The recognizer P system obtained at the end of any computation of the self-constructing stage is* consistent *with the training set.*

The proof of this Theorem is provided in the Appendix A.

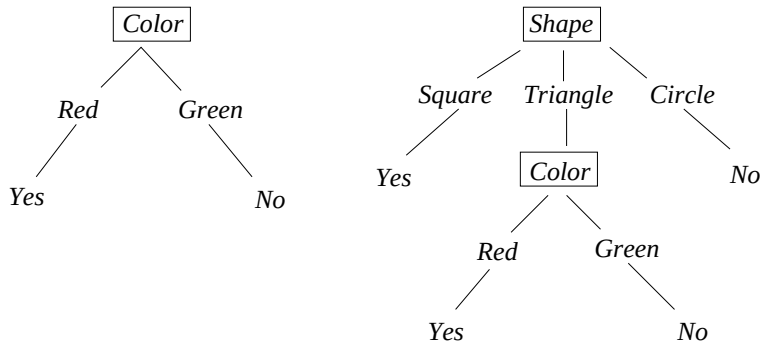


**Fig. 4.** Two trees consistent with the training set from Table 2. Notice that the classification for a green and squared block is different in both trees.

## 4 Conclusions

In this paper, we consider the process of modifying the membrane structure of the membranes as a target itself. This idea can be found in the literature (see, e.g., [3, 6]) but in this paper proposes a new point of view. The new membrane structure (potentially complex), which has been built according P system rules, is the initial membrane structure for a new computation. In this sense, we talk about *self-construction*. This self-construction of P systems can be adapted to many different scenarios and it is independent of the P system model. In such way, this proposal open a new research line, since the adaptation of this idea to different Membrane Computing models need a deeper study.

As an illustrative example, we have considered the well-known problem of constructing decision trees and their use as classifiers in the framework of Membrane Computing. This is also a contribution of this paper, since it provides a new bridge between Membrane Computing and Machine Learning. The purpose has been to illustrate the self-construction of P systems with a simple encoding that allows to translate easily the ideas from Machine Learning into Membrane Computing. In this way, the chosen codification allows a simple description of the process but it is far from being the most efficient. Different codifications will allow more efficient computations and further research must be done in this way.

## Acknowledgements

MAGN acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

## References

1. Ferretti, C., Mauri, G., Zandron, C.: P systems with string objects. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) The Oxford Handbook of Membrane Computing, pp. 168 – 197. Oxford University Press, Oxford, England (2010)
2. Freund, R., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): Membrane Computing, 6th International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised Selected and Invited Papers, Lecture Notes in Computer Science, vol. 3850. Springer, Berlin Heidelberg (2006)
3. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Fractals and P systems. In: Graciani, C., Păun, Gh., Romero-Jiménez, A., Sancho-Caparrini, F. (eds.) Fourth Brainstorming Week on Membrane Computing. vol. II, pp. 65–86. Fénix Editora, Sevilla, Spain (2006)
4. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.J.: On the power of dissolution in P systems with active membranes. In: Freund et al. [2], pp. 224–240
5. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Romero-Campero, F.J.: A linear solution for qsat with membrane creation. In: Freund et al. [2], pp. 241–252

6. Gutirrez-Naranjo, M.A., Perez-Jimnez, M.J., Riscos-Nez, A., Romero-Campero, F.J.: How to express tumours using membrane systems. Progress in Natural Science 17(4), 449–457 (2007)
7. Luisi, P.: The chemical implementation of autopoiesis. In: Fleischaker, G., Colonna, S., Luisi, P. (eds.) Self-Production of Supramolecular Structures, NATO ASI Series, vol. 446, pp. 179–197. Springer Netherlands (1994)
8. Mitchell, T.M.: Machine Learning. McGraw-Hill Education, 1st edn. (1997)
9. Murphy, N., Woods, D.: A characterisation of NL using membrane systems without charges and dissolution. In: Calude, C.S., Costa, J.F., Freund, R., Oswald, M., Rozenberg, G. (eds.) Unconventional Computing, Lecture Notes in Computer Science, vol. 5204, pp. 164–176. Springer Berlin Heidelberg (2008)
10. Mutyam, M., Krithivasan, K.: P systems with membrane creation: Universality and efficiency. In: Margenstern, M., Rogozhin, Y. (eds.) MCU. Lecture Notes in Computer Science, vol. 2055, pp. 276–287. Springer (2001)
11. Pérez-Jiménez, M.J., Riscos-Núñez, A.: A linear-time solution to the knapsack problem using P systems with active membranes. In: Martín-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 2933, pp. 250–268. Springer, Berlin Heidelberg (2003)
12. Păun, G.: Computing with membranes. Tech. Rep. 208, Turku Centre for Computer Science, Turku, Finland (November 1998)
13. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000), see also [12]
14. Quinlan, J.R.: Induction of decision trees. Machine Learning 1(1), 81–106 (1986)
15. Song, T., Wang, X., Zheng, H.: Time-free solution to Hamilton path problems using P systems with d-division. Journal of Applied Mathematics 2013, Article ID 975798, 7 pages (2013)
16. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C., Dinneen, M.J. (eds.) UMC. pp. 289–301. Springer (2000)

## A Appendix

Next, we provide the proof of the Theorem 1. In order to fix the notation, we will call *examples* to the strings $\overline{x}C$, where $\overline{c}$ is a string (maybe empty) of pairs *Attribute Value* and $C \in \{YES, NO\}$. First of all, we will prove the first statement:

> *In the self-constructing stage, in any possible computation, there exists* $p \in \{0, \ldots, k\}$ *such that* $C_{1+5p}$ *is a halting configuration. Therefore, each computation gives at most* $1 + 5k$ *steps.* **(C1)**

In this self-construction stage the strings $\overline{x}?$ are not supplied to the system and, hence $YES_{out}$ or $NO_{out}$ are not produced. In this way, we do not care about the set of rules **R6**, **R7** and **R8** because they cannot be applied. The proof is based on the following lemmas.

**Lemma 1.** *Let us consider an elementary membrane at the step $n$ such that there are no strings in its father membrane (it can also be the skin in the initial configuration) and its content is $YES_{aux}$, $NO_{aux}$, and a non-empty set of examples such that all of them end with $YES$ or all of them end with $NO$. Then, the computation in this membrane finishes at step $n+1$.*

*Proof.* Let us consider that all the examples end with $YES$ (the case $NO$ is analogous). In the described conditions, only one rule can be applied (from **R1**) and in the step $n+1$, the membrane contains the same set of examples and $YES_{act}$ and $NO_{aux}$. It is easy to check that no more rules can be applied and the computation finishes at this step.

From this lemma, we obtain that if all the examples in the initial set provided to the skin in the initial configuration end with $YES$ or $NO$, then the computation of the self-constructing stage end after one computation step.

**Lemma 2.** *All the examples inside a membrane have the same length.*

*Proof.* It is trivial to check that it is true in the initial configuration, since if the training set has $k$ attributes, then, all the initial examples have length $2k+1$. For the next steps, we only need to consider that the rules which sends examples from one membrane into other belongs tho the set **R5**, the application of these rules always decreases the length of the example in two units and all the examples arrive to the membrane simultaneously.

**Lemma 3.** *Let us consider an initial training set without noise. If in a membrane there is at least one example of length 1, then all of them are $YES$ or all of them are $NO$.*

*Proof.* By, Lemma 2, we can consider that all the examples in the membrane have length 1. If the training set has $k$ attributes, then all the examples in the initial configuration have length $2k+1$ and, by construction, these examples of length 1 came from these original one after deleting two objects $k$ times. In this process, if two examples are sent to the same membrane, then both share the same value for one attribute. If both examples are in the same membrane after $k$ deletions, then they share the values of the $k$ attributes and, since their no noise, the classification must be the same.

**Lemma 4.** *Let us consider an elementary membrane $h$ at the step $n$ such that there are no strings in its father membrane (it can also be the skin in the initial configuration) and its content is $YES_{aux}$, $NO_{aux}$, and a non-empty set of examples of length $l$ such that at least one of them ends with $YES$ and at least one of them ends with $NO$. Then, at step $n+5$, the membrane $h$ does not contains strings. It only contains elementary membranes such that contain $YES_{aux}$, $NO_{aux}$ and examples of length $l-2$.*

*Proof.* In the described conditions, at the step $n+1$ the membrane contains the set of examples plus $YES_{act}$ and $NO_{act}$ (by **R1**); at the step $n+2$, it contains the set of examples and *new* (by **R2**); at $n+3$, it contains the set of examples plus $v_1,\ldots,v_s$, where $v_1,\ldots,v_s$ are the values of one of the attributes (by **R3**). By application of rules from **R4**, at the step $n+4$, the membrane contains the set of examples plus $s$ elementary membranes, one for each value. Each of these membranes contains the strings $YES_{aux}$ and $NO_{aux}$. Finally, rules from **R5** are applied and all the examples from the membrane $h$ are sent into the elementary membranes by deleting two objects.

Finally, the Statement 1, can be proved from these lemmas.

*Proof.* Proof of the Statement 1. If the training set has $k$ attributes, then the examples placed in the skin in the initial configuration have length $2k+1$. Two cases are possible:

- If all of them have the same classification, i.e., all of them end with $YES$ or all of them end with $NO$, then, by Lemma 1, the computation finishes in the configuration $C_1$.
- If all of them do not have the same classification, then the conditions of Lemma 2 hold and the configuration $C_5$ has elementary membranes with $YES_{aux}$, $NO_{aux}$ and examples of length $2(k-1)-1$. Each of these membranes is in the same conditions that the skin in the initial configuration, but the length of the examples has decreased in two units. This process goes on and each elementary membrane stops after $1+5p$ steps with $p \in \{0,\ldots,k\}$. Lemma 3 is considered to ensure that all the examples in the elementary membrane end in $YES$ or $NO$ and then, the computation halts as shown in Lemma 1.

Next, we will prove the second statement of the theorem. Now the P system has been self-constructed. Only elementary membranes have strings. As shown in Lemma 1, in the halting configuration, these membrane have a set of examples and the pair $YES_{aux}$ and $NO_{act}$ or $YES_{act}$ and $NO_{aux}$, depending of the classification of the examples. Any computation in the decision stage starts by placing a string $\overline{x}?$ as input in the skin, where $\overline{v}$ is a string of pairs *Attribute Value*. Since all the examples in each membrane have the same classification and no more examples are supplied, then the rules from sets **R1** to **R5** cannot be applied. Only rules from **R6**, **R7** and **R8** must be considered. The statement is the following

*Any computation in the decision stage gives $2p+2$ steps with $p \in \{0,\ldots,k\}$ and sends to the environment $YES_{act}$ or $NO_{act}$ (but no both) in the last step of computation.* **(C2)**

*Proof.* First of all, let us notice that rules from **R6**, **R7** and **R8** cannot be applied simultaneously, because the conditions of applicability are disjoint and only one string $\overline{x}?$ is provided as input in the skin in each computation. From this observation, rules from **R7** are firstly applied $p$ times with $p \in \{0,\ldots,k\}$. After these $p$ steps, the elementary membrane where $YES_{act}$ or $NO_{act}$ is placed and in the

step $p+1$, the object $YES_{act}$ or $NO_{act}$ (but no both) is produced. After $p$ steps more, in the step $2p+1$, $YES_{act}$ or $NO_{act}$ arrives to the skin by application of rules from **R8**. In the following step, $2p+2$, the object $YES_{act}$ or $NO_{act}$ is sent out to the environment and the computation halts.

In the third statement, we consider the self-constructed P system and take an example from the training set $\overline{x}C$, with $C \in \{YES, NO\}$ and replace $C$ by ?. The statement claims that if we provide $\overline{x}$? to the P system in the decision stage, we obtain the same classification that the original one. The statement is

*The recognizer P system obtained at the end of any computation of the self-constructing stage is* consistent *with the training set.* **(C3)**

*Proof.* In order to fix ideas, let us consider that the original example was $\overline{x}YES$. The other case is analogous. By rules from **R6**, $\overline{x}$? will be sent to an elementary membrane where all the examples have the same classification. One of these examples was originally the $\overline{x}YES$ and $YES_{act}$ occurs in this elementary membrane. Then, by application of one rule from **R7** and later with rules from **R8**, the object $YES_{out}$ is sent to the environment.