Decision Trees for Obtaining Active Rules in Transition P Systems

Juan Alberto de Frutos, Luis Fernández, Fernando Arroyo

Dpto. de Lenguajes, Proyectos y Sistemas Informáticos
Escuela Unversitaria de Informática - Universidad Politécnica de Madrid
Crta. de Valencia Km. 7 - 28031 Madrid - Spain
{jafrutos, setillo, farroyo}@eui.upm.es

Summary. The aim of this work is to reduce the time that a membrane spends in working out the active rules subset in every evolution step. With this purpose, it is proposed carrying out a previous static analysis over the Transition P system, obtaning a decision tree with the collected information. In such a way that active rules subset will be determined as a classification problem. It will be shown advantages of incorporating decision trees for this task, and also an analysis of suitability in some architectures proposed to implement Transition P systems. Specifically, architectures based on a cluster of computers and microcontrollers.

1 Introduction

Membrane Computing was introduced by Gh. Păun in [8], as a new branch of natural computing, inspired on living cells. Membrane systems establish a formal framework in which a simplified model of cells constitutes a computational device. Starting from a basic model, Transition P Systems, many different variants have been considered; and many of them have been proved to be, in computational power, equivalent to the Turing Machine. A Transition P System evolves through transitions between two consecutive configurations that are determined by the membrane structure and multisets present inside membranes. It can be considered two sequential phases in every transition step: application of evolution rules inside membranes, and communication among membranes in the system. The present work is focused on optimizing the first one.

The first task in application of evolution rules inside a membrane phase is to determine whether each rule is active or not. The subset of active rules will be applied subsequently in a maximal parallel and non deterministic way. There are many papers in which the main goal is to improve the internal parallelism of the membrane, in such a way that several active rules can be applied simultaneously. However, as regards optimization of the process of obtaining active rules, only the work carried out by Fernández et al. [4] can be mentioned. The main goal of the

present work is to propose decision trees as an optimized solution for determining active rules in some architectures.

Architectures based on a cluster of computers connected by a local net [9], [3], [10], [1] and [2]. Each computer houses several membranes, reaching a certain degree of parallelism. We want to point out the analysis carried out in [10] in which Tejedor et al. try particularly to tackle the bottleneck communication problem, proposing an architecture that avoids communication collisions and reduces the number and lenght of external communications. They conclude that "if it is possible to make that application time be N faster times [...] the number of membranes that would be run in a processor would be multiplied by $\sqrt{N}$, the number of required processors would be divided by the same factor and the time required to perform an evolution step would improve approximately with the same factor $\sqrt{N}$". The goal of the present work fits just in this context, we will try to reduce the application time inside a membrane.

Architectures based on microcontrollers. This line of implementing P systems has been proposed by Gutierrez et al. in [6] and [7]. It consists in a low cost hardware based on microcontrollers PIC16F88 that making use of external memory modules is able to solve the problem of small capacity of storage in these devices. It means a flexible solution due to microcontrollers allow to be software programmed. Figure 1 contains a picture with a real implementation. The represented microcontroller has been adapted to perform membrane execution. Besides, it has been designed to be connected up to with 254 additional microcontrollers.

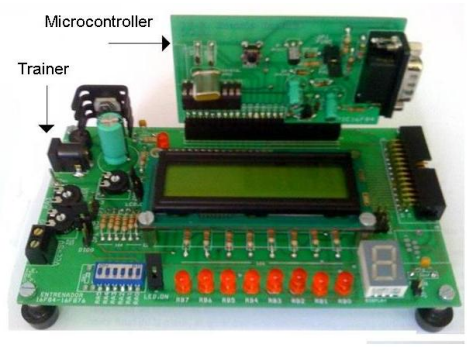

Fig. 1. Circuit with a microcontroller PIC16F88 for implementing P systems.

2 Conditions for an evolution rule to be active

An evolution rule in membrane i can be applied in an evolution step if it fulfils three requisites: useful, applicable and active. A rule r_j is useful if all targets are adjacent to membrane i and not inhibited. A useful rule r_j is applicable if its antecedent is included in the multiset of membrane i. Finally, an applicable rule r_j is active if there is no other applicable rule with higher priority.

The usefulness state concept for determining useful rules is going to be an important issue for the present paper. It was introduced in [5]. This state allows any membrane to know the set of child membranes with which communication is feasible, that is to say, adjacent and not inhibited membranes. This set of child membranes constitute the membrane context, which changes dynamically as membranes are dissolved or inhibited in the P system. The set of usefulness states for a membrane i in a Transition P system can be obtained statically at analysis time, as it is detailed in [5].

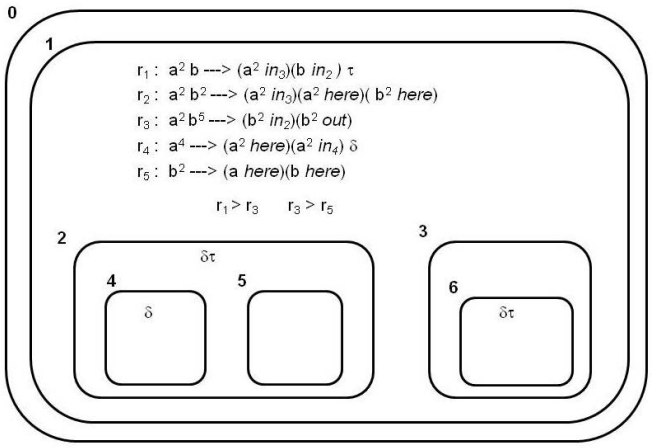

Fig. 2. An example of transition P system

Figure 2 represents an example of a Transition P system. Only rules associated to membrane 1 are detailed. Symbol δ in membranes 2, 4 and 6 represents the possibility of these membranes to be dissolved by application of some rules inside them. The symbol τ represents the possibility of inhibition for membranes 2 and 6 by the same cause. Usefulness states for membrane 1, together with their corresponding contexts, are depicted in first and second columns of table 1. Futhermore, it can be obtained statically the set of useful rules from a given usefulness state q_i. Third and fourth columns in table 1 represent useful rules for every usefulness state.

Usefulness state	**Context**	**Useful rules**	**Useful rules when permeable**
q_0 (1001)	{2, 3}	r_1, r_2, r_5	r_3
q_1 (0111)	{4, 5, 3}	r_2, r_4, r_5	
q_2 (0011)	{5, 3}	r_2, r_5	
q_3 (0001)	{3}	r_2, r_5	

Table 1. Usefulness states for membrane 1

Usefulness states are proposed to be encoded in [5] by means of the **total context** of a membrane, defined as the set of all membranes that eventually can become children of that membrane. For instance, in our example of figure 1, $TC(1) = \{2, 4, 5, 3\}$. Each one of the usefulness states for a membrane i is encoded by $TC(i)$, depending on its context, with binary logic. Thus, the usefulness state q_0, that represents the context {2,3}, is encoded as 1001.

3 Decision trees for active rules

A decision tree is a tool that allows to determine the class which one element belongs to, depending on the values of some attributes or properties of the element. Each non-leaf node of a decision tree corresponds to an input attribute, and each arc to a possible value of that attribute. A leaf node corresponds to the expected value of the output attribute, that is to say, the element classification. An element is classified by starting at the root node of the decision tree, testing the attribute specified by this node and moving down the tree branch corresponding to the value of the attribute. This process is repeated until a leaf node is reached. There are a lot of algorithms to generate decision trees. Specifically ID3 is an outstanding algorithm belonging to TDIDT family (Top-Down Induction of Decision Trees).

Fernández et al. in [4] proposed incorporating decision trees in the calculus of evolution rules applicability. They reach an important reduction of the number of checks necessary for determining the applicable rules subset. The present work, supporting in usefulness states analysis, tries to extend those decision trees, in such a way that conditions for usefulness and priorities among rules will be also taken into account. The decision tree for a membrane will classify the state of that membrane in every evolution step, determining the current active rules subset.

3.1 Attributes

The set of attributes A_i is established as properties necessary to define a state or situation of the membrane i in the P System. Specifically, the set A_i consists of the following attributes:

1. Necessary attributes to set up the usefulness state of membrane i. Thus, there will be one attribute for each membrane belonging to membrane i total context. The associated value will be true if the represented child membrane belongs to the current usefulness state.

$$A_i \supset \{a \equiv m_j \mid j \in TC(i)\}$$

2. One attribute more to determine inhibition in the permeability state of membrane i. The value true corresponds with membrane inhibition.

$$A_i \supset \{a \equiv I\}$$

3. Furthermore, as proposed in [4], we consider attributes for applicability of rules. These attributes represent the set of weight checks between objects from the membrane multiset and objects from antecedents of evolution rules. Neither repetitions nor checks with zero are considered.

$$A_i \supset \{a \equiv |\omega|_u \geq k \mid \ |input(r)|_u = k \ \wedge \ k \neq 0 \ \forall u \in U\}$$

Where $|\omega|_u$ represents the weight of the symbol u in ω (multiset of membrane i); and $|input(r)|_u$ is the weight of the symbol u in the antecedent of r.

As every possible membrane situation will be considered with an instance, the amount of instances in the training data for a membrane i is the following:

$$|E_i| \ = \ |Q_i| \ * \ \prod_{u \in U} \ (\ |C_i^u| + 1 \ )$$

where $|Q_i|$ is the number of usefulness states for membrane i, and $|C_i^u|$ is the number of different checks with symbol u in any rule antecedent of membrane i, that is to say, attributes with the form $|w|_u \geq k$. Besides, if membrane i has inhibiting capability, this value has to be multiplied by 2 in order to consider attribute I.

E	m_2	m_4	m_5	m_3	I	$\|w\|_a \geq 4$	$\|w\|_a \geq 2$	$\|w\|_b \geq 5$	$\|w\|_b \geq 2$	$\|w\|_b \geq 1$	C
$1001Ia^0b^0$	Yes	No	No	Yes	Yes	No	No	No	No	No	∅
$1001Ia^0b^1$	Yes	No	No	Yes	Yes	No	No	No	No	Yes	∅
$1001Ia^0b^2$	Yes	No	No	Yes	Yes	No	No	No	Yes	Yes	∅
$1001Ia^0b^5$	Yes	No	No	Yes	Yes	No	No	Yes	Yes	Yes	∅
$1001Ia^2b^0$	Yes	No	No	Yes	Yes	No	Yes	No	No	No	∅
$1001Ia^2b^1$	Yes	No	No	Yes	Yes	No	Yes	No	No	Yes	$\{r_1\}$
$1001Ia^2b^2$	Yes	No	No	Yes	Yes	No	Yes	No	Yes	Yes	$\{r_1,r_2\}$
$1001Ia^2b^5$	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes	Yes	$\{r_1,r_2\}$
$1001Ia^4b^0$	Yes	No	No	Yes	Yes	Yes	Yes	No	No	No	∅
$1001Ia^4b^1$	Yes	No	No	Yes	Yes	Yes	Yes	No	No	Yes	$\{r_1\}$
$1001Ia^4b^2$	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	$\{r_1,r_2\}$
$1001Ia^4b^5$	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	$\{r_1,r_2\}$
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....

Fig. 3. Instances of membrane 1 for ID3 algorithm

Coming back to the example of P system introduced in figure 1, attributes for membrane 1 are the following:

- As $TC(1) = \{2, 4, 5, 3\}$, four attributes are needed: m_2, m_4, m_5, m_3.
- As membrane 1 has inhibiting capability, an attribute I has to be included.
- Finally, the different checks for applicability in evolution rules are the following: $|w|_a \geq 4$, $|w|_a \geq 2$, $|w|_b \geq 5$, $|w|_b \geq 2$ and $|w|_b \geq 1$.

The resulting training data for membrane 1 are shown in figure 3. As example, $1001Ia^2b^5$ represents the instance in which usefulness state is 1001, permeability state is inhibited, the amount of objects a is at least 2, but not more than 4, and finally, the amount of objects b is at least 5.

3.2 Classification

Each instance is classified into the corresponding set of active rules, as it is shown in figure 3. This task is carried out at analysis time as follows:

- Firstly, useful rules are obtained from the usefulness state and the permeability state (table 1). For instance, $1001Ia^2b^5$ corresponds with the set of useful rules $\{r_1, r_2, r_5\}$.
- Secondly, attributes related to checks of objects weights in multiset determine the applicability property of every useful rule, as it is detailed in [4]. For instance, $1001Ia^2b^5$ corresponds with the set of applicable rules $\{r_1, r_2, r_5\}$, due to every useful rule is also applicable.
- Lastly, priorities among rules have to be considered in order to get the active rules subset, which implies to determine the maximal over the priority relation of the applicable rules subset. With this aim, we have to work out a transitivity matrix (M) expressing the priority relation. Then the maximal is obtained as follows:

$$C = Max(\text{Applicable}) = \text{Applicable} \wedge \neg(\text{Applicable} * M)$$

Following with our example $1001Ia^2b^5$, two priorities are defined for membrane 1: $r_1 > r_3$ and $r_3 > r_5$, which determine a transitivity matrix M. If we represent the applicable rules subset with binary logic as 11001, then $C = Max(11001) = (11001) \wedge \neg((11001) * M) = 11000$, representing $\{r_1, r_2\}$.

Therefore, all instances are available at analysis time. Thus ID3 algorithm can be applied obtaining a decision tree. Specifically, decision tree corresponding to membrane 1 of our example is depicted in figure 4.

Such decision tree can be easily software implemented. Besides, as computation of active rules is a process performed inside every membrane in every evolution step, we propose optimizing it with an assembly language. Anyway, this sofware is a suitable solution for architectures based on a cluster of processors, such as [9], [10], [1] and [2], in which each membrane evolves in a single process. Such process would contain the software obtained for the decision tree. As regards architectures based on microcontrollers PIC16F88, decision trees solution fits properly, due to microcontrollers can be software programmed. More precisely, in [7] Gutierrez et al. made use of the microchip MPLAB IDE integrated environment, in which the tool MPASM allows to work with assembly code. An additional advantage is the avoidance of the transitivity matrix, which is important due to the problem of scarce memory for data in microcontrollers.

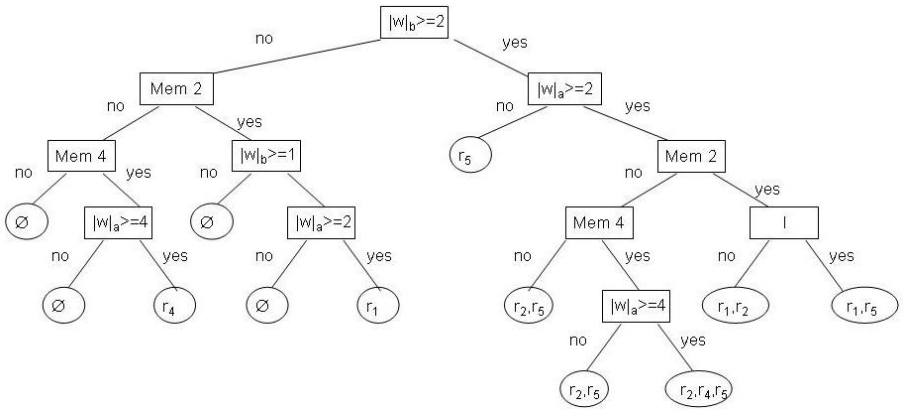

Fig. 4. Decision tree obtained by ID3 algorithm for membrane 1

4 Analysis of results and conclusions

Now, we are going to compare decision trees as proposed here with classical solutions for obtaining active rules, which consist of three sequential algorithms, determining useful, applicable and active evolution rules respectively. Table 2 summarizes the results obtained in this comparison, where $|R|$ represents the number of evolution rules in the membrane, $|TC|$ is the length of the membrane total context, $|U|$ is the amount of symbols in the alphabet and $|C_i^u|$ is the number of different checks for applicability.

	Useful rules	**Applicable rules**	**Active rules**
Classical Algorithms	$O(n)$ $n = \|R\| * (\|TC\| + 1)$	$O(n)$ $n = \|R\| * \|U\|$	$O(n^2) + O(n)$ $n = \|R\|$
Decision tree	$O(n)$ $n = \|TC\| + 1 + \sum_{u \in U} \|C_i^u\|$		

Table 2. Complexity order of algorithms for obtaining active rules

From this comparison, we conclude that decision tree solution performs a fewer number of operations than classical solutions. Additionally, we have applied both kind of solutions to a set of published P systems and results confirm our conclusion. Specifically, the total amount of operations in decision trees vary from 11,32% to 21,21% of the total amount of operations in classical algorithms.

As regards memory requirements, we have to point out some remarks. A decision tree handicap is the need to keep a different code for every membrane. On the other hand, a decision tree advantage is the avoidance of the transitivity matrix. Finally, we have to mention that decision trees could grow up significantly when the number of attributes is high. Then, depending on the implementation architecture, memory space could be insufficient for decision trees and consecuently

classical algorithms had to be chosen. Anyway, as decision tree is obtained at analysis time, the best solution can be determined at that time.

As conclusion, decision tree solution is significantly more efficient than classical solutions for determining active rules. Moreover, some architectures for implementing P systems can get profit from this proposal, such as architectures based on a cluster of computers and architectures based on microcontrollers. Finally, according with the work carried out by Tejedor et al. in [10], this solution means improvements on some architectures proposed to tackle the communication bottleneck problem, such as reduction of the total time of an evolution step, increase of the number of membranes that could run on a processor and reduction of the number of processors.

References

1. G. Bravo, L. Fernández, F. Arroyo. J. Tejedor, *Master Slave Distributed Architecture for Membrane Systems Implementation*, 8th WSEAS Int. Conf. on Evolutionary Computing (EC'07), June 2007, Vancouver, Canada.
2. G. Bravo, L. Fernández, F. Arroyo, M. A. Pea, *Hierarchical Master-Slave Architecture for Membrane Systems Implementation*, 13th Int. Symposium on Artificial Life and Robotics (AROB '08), Feb 2008, Beppu, Oitia (Japan).
3. G.Ciobanu, W. Guo, *P Systems Running on a Cluster of Computers*. Workshop on Membrane Computing (Gh. Păun, G. Rozemberg, A. Salomaa Eds.),2004, LNCS 2933, Springer, 123-139
4. L. Fernández, F. Arroyo, I. García, G. Bravo, *Decision Trees for Applicability of Evolution Rules in Transition P Systems*, Fourth Intern. Conf. Information Research and Applications (i. TECH 2006) June 2006, Varna, Bulgary.
5. J. A. Frutos, L. Fernández, F. Arroyo, G. Bravo, *Static Analysis of Usefulness States in Transition P Systems*, Fifth International Conference, Information Research and Applications (I.TECH 2007), June 2007, Varna, Bulgary. 174-182.
6. A. Gutierrez, L. Fernández, F. Arroyo, V. Martínez, *A Design of a Hardware Architecture based on Microcontrollers for the Implementation of Membrane Systems*, SYNASC 2006, Timisoara.
7. A. Gutierrez, L. Fernández, F. Arroyo, S. Alonso, *Hardware and Software Architecture for Implementing Membrane Systems: A Case of Study to Transition P Systems*, The DNA Inter. Meeting on DNA Computing (DNA13), June 2007, Memphis, USA.
8. Gh. Păun, *Computing with Membranes*, Journal of Computer and System Sciences, 61(1), 2000, 108-143.
9. A. Syropoulos, E.G. Mamatas, P.C. Alliomes, K.T. Sotiriades, *A Distributed Simulation of P Systems*. Workshop on Membrane Computing, Tarragona, Spain, 455-460.
10. J. Tejedor, L. Fernández, F. Arroyo, G.Bravo, *An Architecture for Attacking the Bottleneck Communication in P Systems*, 12th Int. Symposium on Artificial Life and Robotics, Jan 2007, Beppu, Oita, Japan, 500-505.