RTCS: a Reactive with Tags Classifier System

A. SANCHIS, J. M. MOLINA and P. ISASI
Departamento de Informática, Universidad Carlos III de Madrid, Avda, Universidad 30, 28911-Leganés, Madrid, Spain; e-mail masm@inf.uc3m.es

J. SEGOVIA
Departamento de Lenguajes y Sistemas, Facultad de Informática, UPM Campus de Montegancedo, Boadilla del Monte, Madrid, Spain

(Received: 24 December 1998; in final form: 1 September 1999)

Abstract. In this work, a new Classifier System is proposed (CS). The system, a Reactive with Tags Classifier System (RTCS), is able to take into account environmental situations in intermediate decisions. CSs are special production systems, where conditions and actions are codified in order to learn new rules by means of Genetic Algorithms (GA). The RTCS has been designed to generate sequences of actions like the traditional classifier systems, but RTCS also has the capability of chaining rules among different time instants and reacting to new environmental situations, considering the last environmental situation to take a decision. In addition to the capability to react and generate sequences of actions, the design of a new rule codification allows the evolution of groups of specialized rules. This new codification is based on the inclusion of several bits, named tags, in conditions and actions, which evolve by means of GA. RTCS has been tested in robotic navigation. Results show the suitability of this approximation to the navigation problem and the coherence of tag values in rules classification.

Key words: behavior, classifier systems, genetic learning, reactive systems, robotics.

1. Introduction

This work is centered on classifier systems, proposed by John Holland [1, 4, 7 – 10, 12, 18]. A classifier system is a kind of production system. In general, a production system is a set of rules that trigger others and accomplish some actions [5]. Rules consist of a condition and an action. An action can activate the condition of another rule, and thus, some rules interact with other ones. Classifier systems are parallel production systems while traditional expert systems, generally, are not parallel. In a parallel production system, several rules can be activated at the same time, while in not parallel ones, only one rule can be activated in each action. Together with the parallel activation capacity of rules, CSs have the property of learning rule chains syntactically simple to guide their behavior in changing environments, therefore they are considered as learning systems.

In traditional production systems, the value of a rule with respect to other ones is fixed by the programmer together with an expert or a group of experts. In many problems, this could be impossible or inefficient. Therefore, the relative value of

the different rules becomes one of the key pieces of the information that must be learnt. To facilitate this learning, the CS forces rule to coexist in an information-based service economy. A competition among rules is held, where the right to answer to the activation is going from the highest bidders, which will pay the value of their offers to those rules responsible of their activation. In this way, a middle-man chain is formed that goes from manufacturers (the detectors) to consumers (actions toward the environment).

A classifier system consists of three principal components, which can be considered as activity levels. The first level (action level) is responsible of giving answers (adequate or not) for the resolution of the outlined problem. This level is composed of several rules. These rules are codified in chains of characters over a restricted alphabet. The action level generates a response for a given situation. The appropriateness of the response is measured through a reward received from the environment. The second level (credits assignment) distributes the rewards received by the rules that provide the output among the rules and have contributed to the activation of the final rules (which give the output). As in other reinforcement learning methods, this evaluation can be adjusted by applying a high value reward or payment from the environment, if the solution is profitable, and a punishment or negative value if it is not. In this level, it is not possible, however, to modify the behavior of the system by means of changes in their rules, but it is possible to adjust their values and to establish, to a certain measure, a hierarchy of good and wrong rules. The task of the third level (discovery) is to find a new process that allows the system to discover new solutions. In this level, a Genetic Algorithm (GA) [7] is applied.

In short, CSs are machine learning systems that learn syntactically simple string rules to guide their performance in an arbitrary environment [12] and they work over three fundamental concepts, which are related with the three activity levels:

- The solution of the complete problem is a set of rules (a subset of rules is a solution to concrete situations; even an isolated rule can be a solution for a specific situation, although this is not frequent). Relationship among rules is carried out in several internal cycles.
- Payment of each rule is distributed among rules that activated it in the internal cycles.
- Genetic algorithm allows to generate rules from the better ones, producing, theoretically, an improvement in the global functioning of the system.

The operation form of classifier systems presents some problems in:

1. Execution time.
2. The learning of complex strategies.
3. The definition of the instant to call the GA.
4. The presentation of the examples to the system.

Centering over the first two problems, they are due to the existence of internal cycles. However, these internal cycles allow the interrelationship among rules in

order to produce elaborate solutions. While a CS executes internal cycles, it remains isolated to the environmental information. This problem can be described as the necessity of a CS of being capable "to react" to the stimuli of the environment. The problem to reach the "reactivity" in classifier systems has been approached from two different perspectives:

- Decreasing the time needed to produce an output. This is the solution developed by Dorigo [3] for the ICS and hierarchical CS.
- Executing a rule for each input, without internal cycles and, therefore, without rule sequences, the HCA of Weib [25] based on Wilson [26] and Grefeenstete [6] works.

The problem of the capacity of these systems to learn rule chains appears because rule chains cannot be broken between different learning instants. The loss of a rule of the chain could cause the loss of all the knowledge due to the interrelationships between rules. Isolated rules make no sense, but they are significant in groups, unknown *a priori*. These problems have been approached in the bibliography. Shu et al. [23] solves the problem through the introduction of hierarchies in the CS. The hierarchy defines groups of rules that have been maintained along the learning process. These rule groups are formed *a priori* and they are built by an expert in the solution of the problem. The objective of these authors is to solve the same problem that DeJong solved in genetic algorithms through crowding [4].

In this work, a Reactive with Tags Classifier System (RTCS) has been designed, in the sense of Weib, but without loosing the ability of elaborating complex strategies. For this purpose, it is necessary to remember the definition of reactivity: a reactive system must decide for each input an action, and each action is determined by an input. In a CS the reactivity ought to be considered without losing the capacity of chaining rules in different time instants. To obtain a RTCS, the operation of the action level has been modified. The solution proposed, therefore, must join the capacity of learning without previous knowledge with the capacity of generating some kind of internal subdivision within CS to allow the existence of rule categories. To carry out this solution, the codification of rules should be modified to include a field that represents the type or group to which belongs each rule; this field is named tags.

2. Reactive with Internal Tags Classifier Systems

2.1. REACTIVE WITH TAGS CLASSIFIER SYSTEM ARCHITECTURE, RTCS

A schematic representation of a traditional Classifier System is shown in Figure 1. In this system, three activity levels can be distinguished:

1. **Performance**, also called rule and message system: it interacts with the environment, gathering information through the input interface and producing the output through the output interface; it also receives the payoff. Structurally, the

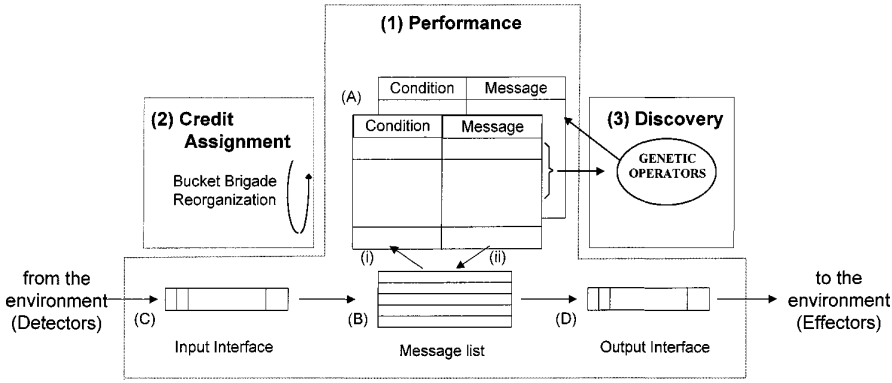

Figure 1. Representation of a traditional classifier system.

performance level consists of: (A) a finite population of fixed length condition/action rules, (B) a message list, (C) an input interface consisting of a set of environmental feature detectors and (D) an output interface for acting in the environment (also shown in Figure 1).

2. **Credit assignment:** it causes rules to be established (fitting a rate of rules) on the basis of their observed utility to the system goal. A reinforcement algorithm, the Bucket Brigade [11], has been employed. This algorithm allows selection from incompatible or contradictory solutions. The Bucket Brigade algorithm assigns to each rule a value, called *strength*, that indicates the usefulness of the rule in respect to the goal. This strength value is also involved in the rules activation process [11].
3. **Discovery:** it employs a genetic algorithm as a discovery mechanism that automatically generates new rules. From a CS, a set of rules with higher strength values is selected, genetic operators are applied and the newly obtained rules are included in a new CS. After this, the Bucket Brigade algorithm will reorganize rule strength values.

In a CS, rules are composed of two parts: condition and message, and they are codified as strings: each condition is a string of fixed length k over the alphabet $\{0, 1, \#\}$ (the "do not care" symbol # matches both 0 and 1) and each message is another string of fixed length k over the alphabet $\{0, 1\}$.

Following this architecture, the performance level has been modified to learn reactions and actions. The performance level is composed of conditions and messages in the same way as a traditional CS except for two main differences:

(1) condition/message length k is longer than the environmental message length m ($k > m$), and

(2) both conditions and messages are divided in three blocks. Each block contains different kinds of information (Figure 2), environmental information, information related with rules fired in a previous instant (internal conditions) and information about decisions.

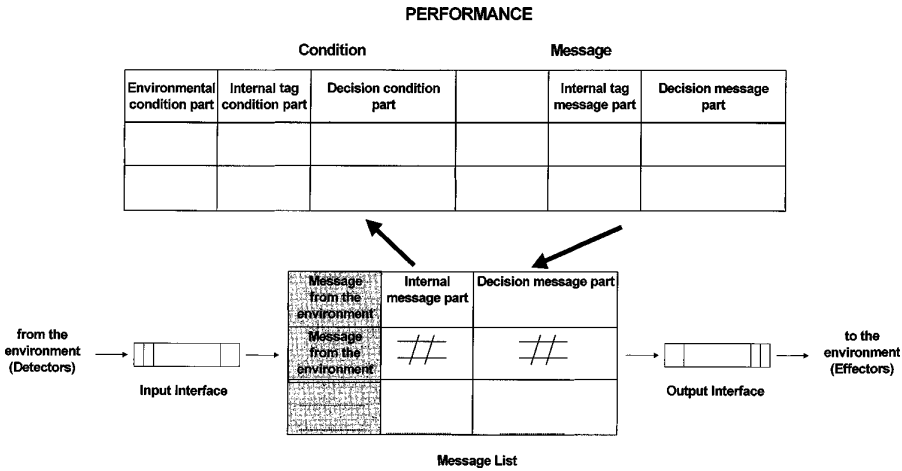

Figure 2. Performance level in an RTCS.

As could be seen in Figure 2, the first block of the message, the environmental block, is empty. This empty block is used to fuse the environmental message with messages of the previously activated rules. The complete sequence of operations will be explained in more detail in Section 2.2. This fusion mechanism allows the RTCS to learn complex actions, composed of a sequence of actions. Besides fused messages, another message with only the first block of the message, the environmental part, is posted to the message list. This last message allows learning reactions, breaking the rules chain.

2.2. SEQUENCE OF OPERATIONS IN A REACTIVE WITH TAGS CLASSIFIER SYSTEM

In a traditional CS, when a codified message arrives from the environment (through the input interface), the message is set in the message list. The message list is compared with all the rules and those that match with some messages are fired. The fired rules post their messages into the message list. Several rules could be activated in parallel by a message. Before rules post messages, the message list ought to be cleaned. Activation of rules is repeated for n cycles. Finally, a message is chosen to give the output through the correspondent interface. The sequence of operations is summarized in Figure 3.

In a RTCS, when a codified message of length m arrives from the environment through the input interface, the message is fused with messages of previously activated rules. Beside, a message composed with the environmental message and "do not care" symbols is posted to the message list. All the rules that match with some messages of the message list are fired. A message is chosen from these fired rules. The list is kept to the next decision cycle. These operations do not contain

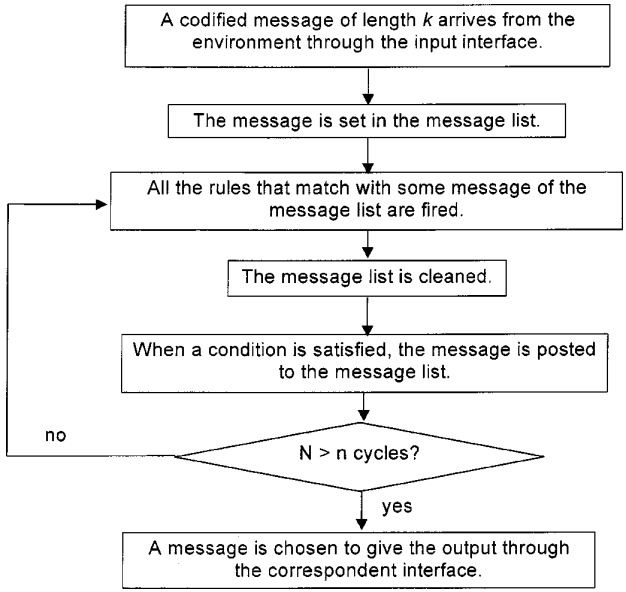

Figure 3. Representation of a sequence of operations in a traditional classifier system.

the repetition of the matching process of the traditional CS, because the chain of rules needs the information of the next environmental input. The rules chain is over different inputs, using internal conditions and message fusion, allowing to learn reactions and actions sequences. The sequence of operations is summarized in Figure 4.

2.3. KNOWLEDGE LEARNED IN RTCS

In this new RTCS, actions are chained taking into account two special mechanisms in conditions and messages: the environmental message is fused with the previously posted messages and internal tags are added to evolve a chaining strategy. This strategy allows to chain rules activated by the environmental message with previous activated rules. The fusion method gives way to chain rules and the internal conditions support the knowledge about the relationship between rules. The evolution process over the internal conditions provided by the genetic algorithm leads to learn sequences of rules through time.

Although all the messages in the message list are composed by fusion, there is always one message with only the environment block filled with the environmental message ("do not care" symbols #, filling the other two blocks, see Figure 2). In this case, the matching process considers only environmental conditions and the system is able to break the rules chain and to react to the environment. This is the way through which reactions are obtained.

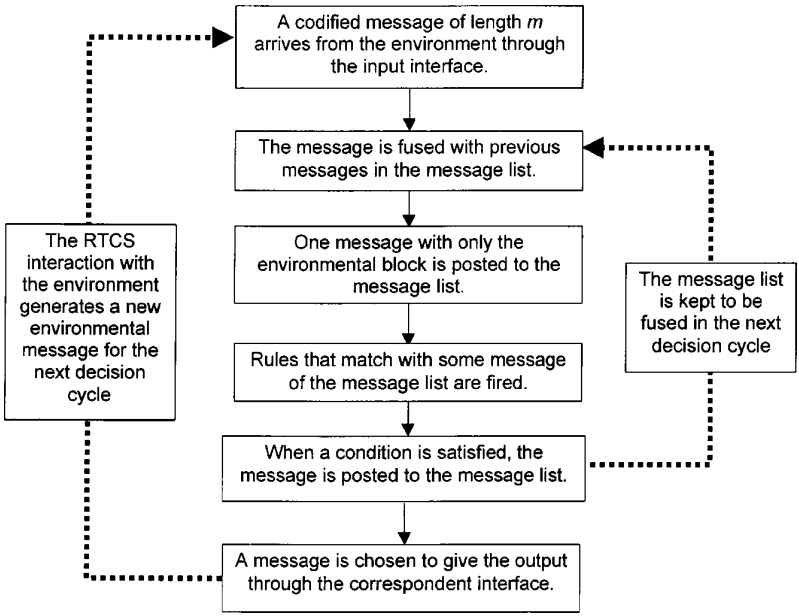

Figure 4. Representation of a sequence of operations in a RTCS.

These two mechanisms allow the generation of more complex rules needed for the final solution of the problem. An example of condition/action rules due to the fusion mechanism that could evolve is as follows:

```
IF External\_Signal IS <type x>  AND
   Last\_Rule\_Fired IS <type y> AND
   Decision\_Part IS <type z>
THEN Send\_Message <001...>
```

The reaction mechanism, on the other hand, allows the generation of traditional reaction rules as:

```
IF External\_Signal IS <type x>
THEN Send\_Message <001...>
```

3. Validation of RTCS

Traditional classifier systems have proven to be appropriate learning systems. However, in many cases, their usefulness is reduced when the environment is dynamical, the problem is real and, furthermore, the behavior to learn is reactive. In this work, the limitations of traditional CS are overcome when RTCS are applied, for example, in an autonomous robotic environment.

A fundamental requirement for autonomous robots is navigation, understanding navigation as a task that allows a robot to move, from place to place without danger neither damages. A classic example of architecture of control is the designated "subsumption", proposed by Brooks [2], that has been implemented with success in robots of MIT and other institutions. The subsumption architecture is based on behavior. Each behavior reacts in a situation and the global control is a behaviors composition. To implement these behaviors different systems have been employed: from finite state machines to fuzzy controllers. These behavior rules could be designed by a human expert, designed "ad hoc" for the problem, or learnt through some artificial intelligence techniques [16]. Some approximations have employed genetic algorithms to evolve fuzzy controllers [14], evolutionary strategies to evolve connections weights in a Braitenberg approximation [13, 19], or neural nets for behavior learning [20].

When a traditional CS is employed for learning reactive behaviors, an additional problem is detected in respect to the actions chains (generated in several internal cycles). These actions chains blind the system, make it insensitive to the environment during the execution of the chain, since the system can not manage any new input during the decision process. If, furthermore, the environment were dynamical, the system would have to read the sensors (input, situation of the environment) in each decision step, since this is the principal characteristic of reactive systems. For example, in the navigation of an autonomous robot through a dynamical environment problem studied (where the obstacles can be mobiles), the robot would not have to remain blind any moment. Therefore, each movement must be the result of the application of a decision process over the last reading of sensors. To solve this problem, the Reactive with Tags Classifier System, described in Section 2, RTCS, is proposed [21, 22].

In the proposed learning system, the only previous system information is related to the number of inputs (in the robot will be number of sensors), the domain, the number of outputs (in the robot, number of motors) and their description. Thus, the robot controller (the RTCS) starts without information on correct associations between sensors inputs and motors velocities. From this situation, the system (robot + controller) must be capable of learning to reach the greater degree of adaptation to the sensors information.

The robot has to discover a set of effective rules, employing past situations experience, and must extract information from each situation, when this is produced. In this way, the system will learn in an incremental way and past experiences remain implicitly represented through evolved rules.

3.1. INPUT AND OUTPUT CODIFICATION

The codification of information in CS (the design of environmental and output messages) is based on the special problem where CS will be applied. In this work, the CS is used as a controller of an autonomous robot named Khepera [20]. The

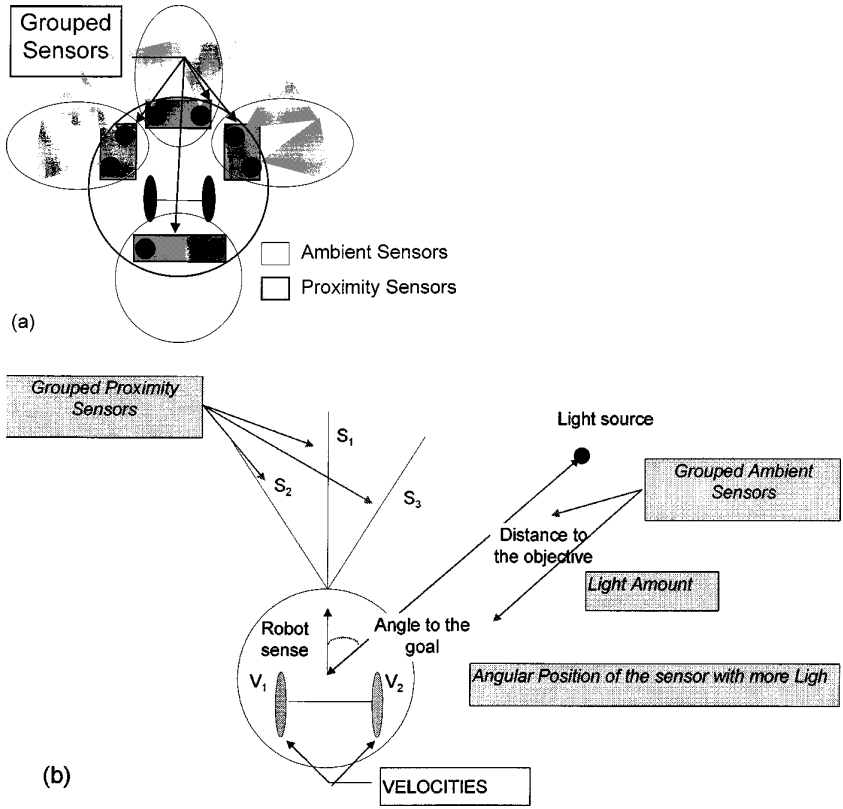

Figure 5. (a) Sensors considered in the real robot. (b) Input information to the system.

sensory inputs come in from eight infra-red proximity/ambient sensors. The robot has two wheels controlled by two independent DC motors with an incremental encoder that allow any type of movement. Each wheel velocity could be read through a speedometer.

The sensors (proximity, ambient and speedometer) supply three kinds of incoming information: proximity to the obstacles, ambient light, and velocity. Instead of using the eight infra-red sensors individually, they have been grouped giving a unique value from two sensor input values (Figure 5(a)). In this sense, the amount of information received by the CS is reduced as well as the length of rules. The robot goal is a light source, the ambient information lets the robot know the angle (the angle position in the robot of the ambient sensor receiving more light) and the distance (the amount of light in the sensor) (Figure 5(b)).

The input to the RTCS codified forms the environmental message. This message consists of three proximity sensors, angle and goal distance (given by ambient sensors) and velocity values obtained by the speedometer. The outputs are the velocity values. The composition of the message can be seen in Figure 6.

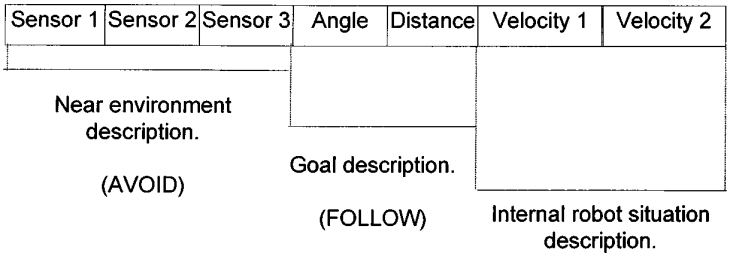

Figure 6. Composition of the environmental message.

Table I. Codified values of proximity, angle, distance and velocities

Codification	Proximity	Angle	Distance	Velocities
00	Very Near (VN)	Near 0 (0)	(0, 25) (VN)	Slow Forward (F)
01	Near (N)	$<\pi$ (0–PI)	(25, 100) (N)	Fast Forward (FF)
11	Far (F)	$>\pi$ (PI–2PI)	(100, 200) (F)	Backward (Bc)
10	Very Far (VF)	Near 2π (2PI)	(200, ∞) (VF)	Stop (ST)

The distance information of proximity sensors is obtained by the response curve of the sensors. The distance domain is transformed, translating it into a simpler domain to codify the values. This transformation allows both the CS and the robot to be independent. So, the CS could be developed for any robot by changing the transformation function. The maximum distance value "seen" by one sensor is 40 units and is divided in four equal sets. The angle sets are of different size to consider a fine fitting of the trajectory, avoiding big oscillations when the robot follows the right direction (the sets near 0 and 2π are smaller than the "$<\pi$" and the "$>\pi$" ones). To keep the independence between robot and CS, the distance values are translated from the real sensor values to a domain defined from 0 to ∞. The input domain has been partitioned in four sets. Velocity values flow as input to the CS and as decision from the CS to the robot. The values are defined by the maximum and minimum velocities $(10, -10)$. This range is divided in four equal sets. All these sets should be codified to build the message from the environment. Two binary digits are needed to represent each set. The codified inputs to the robot are displayed in Table I.

3.2. PAYMENT FUNCTION ADJUSTMENT

In a navigation problem, those actions that permit the robot not to collide will be considered as positive (for example, the actions to increase the distance to some obstacle, or to approach the goal, or the alignment of the robot in the goal direction). Those actions that remove it from the goal, for example, or those that increase

the distance traveled by the robot, or that brings it closer to the obstacles and may produce some collision can be deemed as negative. So, the considered factors to calculate the reward are: increase of the distance to an obstacle, approximation to the objective and alignment or draft toward the objective.

The function chosen to calculate the payment is given by Equation (1), that will constitute the final payment through Equation (2). Different constants are employed to obtain the adequate influence of each one of the factors and without distorting strengths values.

$$P_s = P_1 \times \text{Coef}_1 + P_2 \times \text{Coef}_2 + P_3 \times \text{Coef}_3, \quad (1)$$

$$P_T = P_s \times K_s, \quad (2)$$

where:

- P_1 is the corresponding part to the approximation to the objective. Its value comes determined by the difference between the distance in the previous execution cycle and the current distance.
- Coef_1 is a constant applied on P_1.
- P_2 is the corresponding part to the alignment toward the objective. Its value determines the difference between the angle in the previous cycle and the current angle, in radians, being positive if turned toward the objective and negative otherwise.
- Coef_2 is a constant applied to P_2.
- P_3 is the corresponding part to distance to objects. It is calculated evaluating left and right sensors (S2 and S3). If the value of S2 is less than S3 value, it is paid turning to the right, and if it is the opposite, it is paid turning to the left. If the turn is wrong, it is penalized in the same quantity. If S2 and S3 are equal, neither is paid nor penalized.
- Coef_3 is a constant applied to P_3.
- P_s collect the result of previous payments.
- K_s is a constant applied on P_s.
- P_T is the final payment or reward.

A collision with an object is not included in the previous function. In this case, a punishment greater than any other case is applied, with a fixed value of P_s since, as there is no movement, there is no evaluation neither turns nor approximations.

Fitting and correct election of this function will determine the success of this kind of systems; so it would be necessary to include in a RTCS all rules, containing all possible conditions and all possible actions (messages) for each condition. Thus, for each possible condition 16 different messages are generated (speed of each wheel is codified with 2 bits, then 2^4). Once all the possibilities have been taken into account, when executing the RTCS, strength values of better rules would have to increase and reduce the values of the worse. At the end of execution, each one

of the rules obtains a value of strength that reflects, in a real way, their usefulness in the system.

An important problem related to the number of necessary rules to reproduce all the possible situations appears. This number is calculated considering the number of bits involved in each possibility. It will be n the total number of necessary rules: $n = comb\_\text{S1S2S3} \times comb\_AngDist \times comb\_V1V2dec$, being $comb\_\text{S1S2S3} = 2^6$, $comb\_AngDist = 2^4$ and $comb\_\text{V1V2}dec = 2^4$. This produces an $n = 2^{14} = 16.384$ different rules. This number of rules is excessively high and impossible to handle. That makes it necessary to appeal to some other method of adjusting the reward function.

The proposed solution is to divide the rules of the RTCS in two groups: one for following rules and the other for avoiding ones. Rules of the following group will have, in their conditions, different values in the part of "following" (angle and distance sensor) and the rest with symbol #. Similar codification is adopted for "avoiding". Thus, the number of rules of the RTCS would be $n = (comb\_\text{S1S2S3} + comb\_AngDist) \times comb\_\text{V1V2}dec$. This corresponds to 1.280 different rules and continues being an excessively high number of rules. Therefore, those classifiers whose conditions will be redundant have been eliminated. For example, if one of the sensors perceives that there is an obstacle very near, 00 in the corresponding position, values of other sensors, distance and angle lose importance since results necessary to turn in opposite sense, and be removed from the object.

In this way, 544 classifiers are eliminated. Sensors S2 and S3 detect obstacles present to the left and right of the robot, respectively; if some obstacle appears in those positions, it is necessary to avoid the obstacle, independently of the rest of the values. This reduces the number of rules by 192. In respect to the following part in conditions, it is only possible to reduce the aspect that considers the minimal distance to the objective, without considering the angle, since it is considered that the objective has been already reached. That reduces rules number in 48.

Finally, rule number will be: $n = 1280 - 544 - 192 - 48 = 496$. It is not possible to reduce the number of rules more and however, 496 continues being an excessively high number of rules for an RTCS applied to a reactive problem. So, four RTCS, each one containing 124 rules, are used in a competition among them. It is necessary to assure that for each possible rules condition, all the possible movements of the wheels are represented. Thus, for each condition the following actions are fixed, as it could be seen in Table II.

Finally, trying to obtain a generalized solution, 15 executions are accomplished over each one of the RTCS, with three different initial situations of the robot (Robots 1–3, summarized in Table III). The process is repeated 5 times, to obtain 15 executions.

The competition among all possibilities is held in order to adjust the function. After one execution, a new RTCS is generated, containing the better rules of the previous RTCS. The competition process is defined in the following way: once executed RTCS1, RTCS2, RTCS3 and RTCS4 during 15 executions (5 of each one

Table II. Velocity values in the four RTCS

RTCS 1		RTCS 2		RTCS 3		RTCS 4	
V1	V2	V1	V2	V1	V2	V1	V2
5	0	5	–10	5	5	5	10
10	5	10	10	10	–10	10	0
0	–10	0	0	0	10	0	5
–10	10	–10	5	–10	0	–10	–10

Table III. Initial values of the three robots

Robots	X	Y	Sense
Robot 1	50	400	0
Robot 2	300	450	180
Robot 3	50	150	0

for 3 different initial situations of the robot), better actions for each condition of each RTCS are chosen (those with greater values of strength). Thus, RTCS5 and RTCS6 will be obtained. Repeating the process with these two new RTCS, finally, RTCS7 is obtained, which contains the better rules. Finally, from RTCS7, RTCS8 is generated, choosing the better two rules for each possible condition. RTCS8 contains 62 rules. In Figure 7 a schema of selection processes of the RTCS for the adjustment of the payment function is shown.

The Equation (2) is re-defined by Equation (3), where the parameters are empirically obtained.

$$P_s = P_1 K_1 C_1 + P_2 K_2 + P_3 K_3 C_3, \tag{3}$$

where:

- P_1, P_2, P_3 and P_s are previously described parameters in Equation (2).
- K_1 is a fixed value applied on P_1.
- C_1 is a variable value applied on P_1, function of the near to the objective. Its values are: 4 if the robot is very near to the objective, 2 in the intermediate case, and 1 if it is far of the objective.
- K_2 is a fixed value applied on P_2.
- K_3 is a fixed value applied on P_3.
- C_3 is a fixed value applied on P_3 whose value depends on the distance to objects. Its value is 8 if it is very near to the object to avoid, 4 in the following distance section, and 1 in the rest.

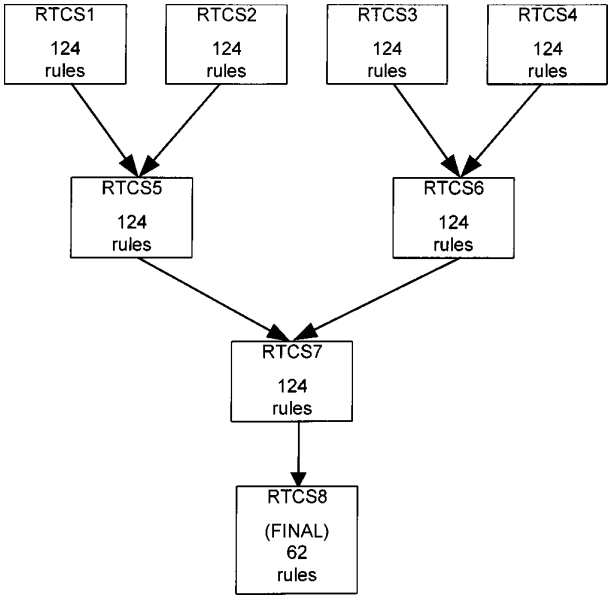

Figure 7. RTCS selection process schema.

Table IV. Empirical values of constants

K_1	K_2	K_3	K_s	P_s
0.3	1	3	0.02	–10 (if collision happens)

The obtained values for the function constants are summarized in Table IV.

Despite the fact that these parameter values have been empirically calculated, these values reflect the importance of each contribution. Thus, the greater, and therefore, more important value, has to deal with avoiding the obstacles. This value is ten-fold greater than the corresponding factor to contribution of being aligned with the objective. Though the approximation factor to the objective could seem smaller, the angle magnitude is less than those of traveled distance. Furthermore, this difference of magnitude is taken in consideration through the constant C_1, according to where the robot is found. Thus, when the robot is near the objective, C_1 causes the coefficient for the approximation (Coef_1) to be equal to the alignment one (Coef_2). When the robot is far away from the objective, both coefficients (specified by K_1 and K_2) keep the relationship of magnitudes.

Finally, to analyze strength values evolution of the rules by means of the payment function, an RTCS has been generated, named RTCS9, containing 124 rules, 62 of which belonging to RTCS8 and 62 randomly generated. In Figure 8 strength variations of some of these rules are shown, during the execution of RTCS9. As

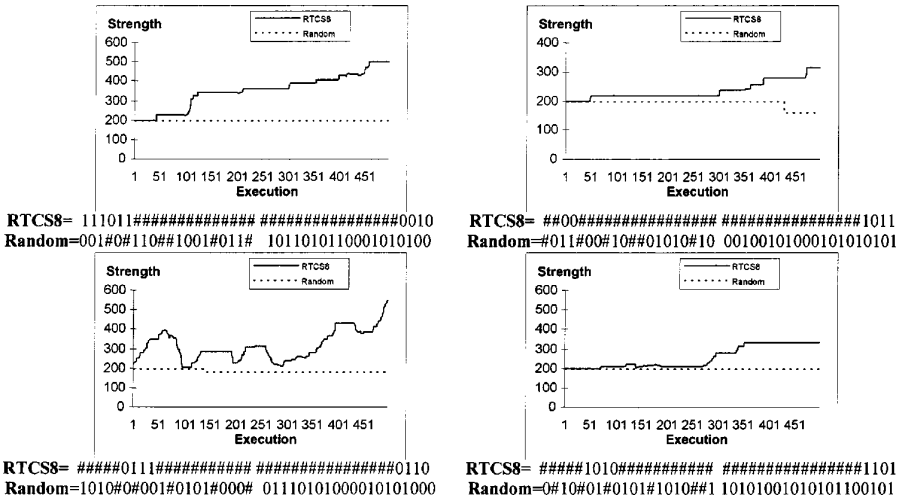

Figure 8. Comparison among strength evolution in 4 randomly generated rules and 4 selected from RTCS8 during RTCS9 execution.

can be observed rules that have been randomly generated are poorer than the "ad hoc" ones, therefore values of their strength stay constant if they are not activated, or decrease, if they are activated.

3.3. LEARNING WITH RTCS

Experiments take a long time of continuous functioning of the hardware. In order to prove the different configurations of CSs, both traditional and RTCS, a simulator, the SimDAI, developed in the previous work [24] has been used. In the simulator, the characteristics of the turtle robot model [17] and the physical restrictions of the Khepera robot have been considered. SimDAI is a working prototype of a mobile robot simulation environment for experimenting with robot navigation and control algorithms. Each mobile robot is completely independent, it can navigate and interacts with other robots in a 2-D simulated world of obstacles, which is separately monitored. This simulator has been used in many other works [13, 15, 19, 21, 22].

The simulation world consists of a rectangular map of user defined dimensions where particular objects are located. In this world it is possible to define a final position for the robot. In this case, the robot is represented with three proximity sensors and two special sensors to measure the distance and the angle to the goal (Figure 9, of a real environment Figure 10).

Evaluation of the system performance is based on a quantitative measure. This measure does not take part in the evolution process but it reflects the system's global performance evolution. For measuring system evolution, the following features have been considered:

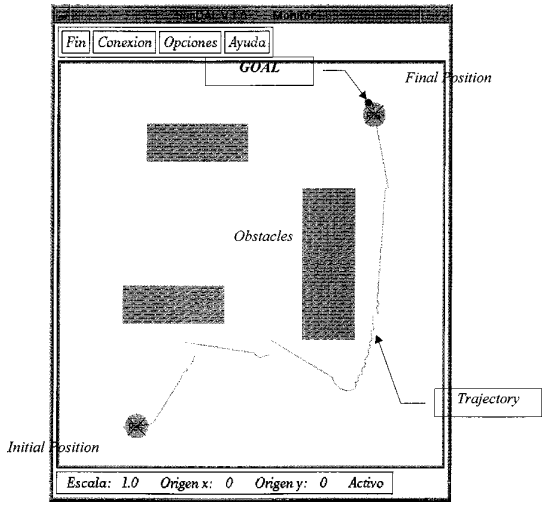

Figure 9. SimDAI simulator (example of one simulated environment).

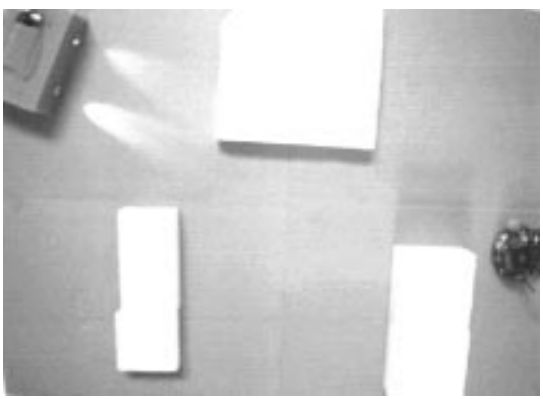

Figure 10. Example of a real experimental environment.

- Time needed to reach the goal (seconds in the real robot and cycles in the simulator).
- Trajectory length (measured by means of velocity values of the motor wheels).
- Number of collisions (measured using the minimum value of the proximity sensors).

In these experiments, the initial population of the RTCS is randomly generated. In this case, the ability and improvement of the RTCS to learn reactions compared with the traditional approach can be probed. The parameters of the CS, traditional and new, are equal:

Figure 11. Global evaluation of the two systems.

- the GA is called after 100 cycles of decisions,
- 1 of crossover probability,
- 0.01 of mutation probability,
- 0.3 of overlapping.

Four internal cycles in the performing level are considered in the traditional CS [4].

The simulator executes the robot controller like in the real world, so, while traditional and reactive CSs take a decision, the robot is continuously working. The velocity of the robot in this period is the previously decided velocity. This velocity is changed when the CS takes a decision for the incoming environmental message. This consideration takes a main place in the traditional CS because it executes four internal cycles before taking a decision. Figure 11 shows the evolution of a function that linearly combines the time and distance is shown (the function is: $1{,}5\times$ time $+$ distance).

The achieved rules in RTCS improve the performance of the robot by 60% compared to the rules obtained with the traditional CS.

4. Real Robot Experiments

The experiments were accomplished in order to compare the results obtained by the same classifier system both in the simulator and in a real environment, with the Khepera robot. The environment consists of several elements:

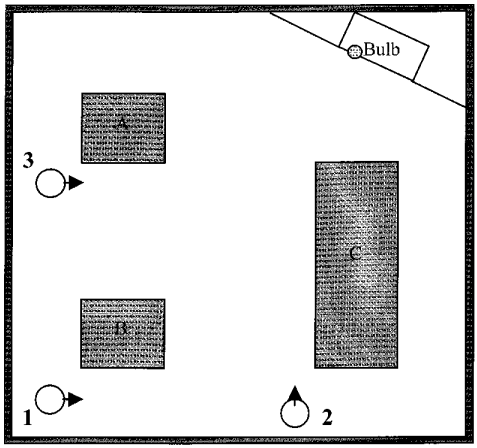

Figure 12. Schema of real robot experiments.

- A wood enclosure in white color, of 6.5 cm of high, and 70 × 70 cm of perimeter.
- A bulb of 2,5 V placed in a foam chunk and fed by a continuous current generator.
- The surface of the enclosure is covered of a black color cardboard, to assess the optimum behavior of the robot respecting to the source of light.
- Three objects have been placed on the enclosure in a similar way to the simulator world. These objects are of white color, 10 × 10 cm and 6 cm in height.

In Figure 12, a plan of the described real environment is shown. Three different starting positions of the Khepera appear. These positions are also similar to the ones used in the simulator.

Twelve experiments have been accomplished, each one consisting of 20 consecutive robot executions. In the experiments, starting position and robot sense change (positions 1–3), and also objects number objects (A, B and C) with their possible combinations, eliminating an object). In each execution three objective parameters have been collected: the number of produced collisions, time elapsed until arriving at the objective, in seconds, and distance traveled, in centimeters. Furthermore, for each experiment, maximum and minimal values, average values and standard deviations have been calculated, for each one of these three parameters. In Figure 13 some comparative tables are shown.

As can be observed in these obtained data, RTCS on a real robot operates almost without collisions in all situations and reached the goal in a relatively short time (a similar duration). These results demonstrate that learned rules are useful for the navigation of a real robot with a stable and fixed functioning, so, behavior of the

	POS 1 ABC			POS 1 AB			POS 1 AC			POS 1 BC		
	Col	Time	Dis	Col	Time	Dis	Col	Time	Dis	Col	Time	Dis
Maximum	6	3:01	160,3	0	1:05	85,16	4	1:44	119	4	2:36	179,9
Minimal	0	0:55	67,13	0	0:48	64,74	0	0:48	61,80	0	0:52	65,17
Deviation	1,70	0:30	24,44	0,00	0:05	5,93	0,91	0:17	18,74	0,99	0:26	29,51
Average	0,60	1:18	83,51	0,00	0:54	70,04	0,25	1:06	78,38	0,35	1:15	88,08

	POS 2 ABC			POS 2 AB			POS 2 AC			POS 2 BC		
	Col	Time	Dis	Col	Time	Dis	Col	Time	Dis	Col	Time	Dis
Maximum	2	1:55	129,8	0	0:40	49,79	0	2:04	127,1	4	1:59	125,3
Minimal	0	0:39	52,39	0	0:33	46,77	0	0:42	53,31	0	0:44	53,89
Deviation	0,45	0:20	18,93	0,00	0:02	0,95	0,00	0:25	21,74	0,91	0:19	19,12
Average	0,10	0:59	65,57	0,00	0:36	47,85	0,00	1:04	69,50	0,25	1:02	71,78

	POS 3 ABC			POS 3 AB			POS 3 AC			POS 3 BC		
	Col	Time	Dis	Col	Time	Dis	Col	Time	Dis	Col	Time	Dis
Maximum	1	1:04	60,52	0	1:09	81,24	2	1:14	72,57	0	0:38	45,52
Minimal	0	0:32	43,50	0	0:34	43,46	0	0:31	42,04	0	0:31	41,30
Deviation	0,22	0:08	4,73	0,00	0:08	8,12	0,49	0:11	7,88	0,00	0:02	1,33
Average	0,05	0:42	47,28	0,00	0:39	48,74	0,15	0:39	47,66	0,00	0:33	42,97

Figure 13. Results of RTCS in real robot.

RTCS on the real robot demonstrates that the degree of learning of the RTCS is enough to carry out the imposed task.

5. Analysis of Learned Rules in RTCS Applied to Navigation of an Autonomous Robot

Results obtained with RTCS are caused by, on the one hand, the introduction of Internal Tags (IT) and, additionally, the introduction of the mechanism that allows the CS to be reactive. In this section, the influence and contribution of Internal Tags will be analyzed. The rules learned by the RTCS have been carried to the real robot and it has been proven its efficiency in navigation (previous section).

To analyze internal tags values, the number of groups formed are studied. Each group is defined according to the IT value in the condition part of the rule. Each group is formed by a set of rules that share some kind of information. Groups have been automatically generated through the evolutionary process. The meaning of each group is not perfectly clear, but it is possible to observe a similar trend in all rules of a same group. More complex analysis includes the explanation of

Table V. Obtained values of internal tags

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
IT	00	01	11	#1	10	1#

Table VI. Group 1 rules

Condition							Message			
s1	s2	s3	A	d	v1	v2	IT	v1	v2	IT
F or VF	VN or N	VF	0	N	Bc	ST	00	F	F	0#
VF	VN	N or VF	0	VF	F	F	00	F	F	00
VF	VN or L	N or VF	0	VN or F	F	F	00	FF	FF	00
F or VF	F or VF	VN	0–PI	VF	Bc	ST	00	F	F	0#
F or VF	N or VF	VN or N	2PI or 0–PI	VF	FF	FF or Bc	00	Bc	ST	00
VF	VF	VN or L	0–PI	VF	FF or Bc	all	00	Bc	ST	0#
F or VF	VN or N	VF	0–PI	N	FF	FF	00	ST	Bc	#0
VF	VN or N	VF	0–PI	F or VF	ST or F	Bc	00	ST	Bc	#0
VF	VN or N	VF	0–PI	F	F or FF	FF	00	ST	Bc	0#
VF	N or VF	VN or N	2PI or 0–PI	F	F or FF	FF	00	Bc	ST	0#
F or VF	VF	VN	0 or PI–2PI	VF	ST or Bc	ST or Bc	00	Bc	ST	0#
F or VF	VN	VF	2PI or PI–2PI	VF	ST or F	Bc	00	F	F	11
VF	VN or N	VF	PI–2PI	N or VF	ST or Bc	ST	00	F	F	0#
VN	F or VF	VF	0 or PI–2PI	F or VF	Bc	ST	00	Bc	ST	0#
VN or N	VF	VF	0–PI	VF	all	ST	00	F	F	0#
VN or N	VF	VF	2PI	N	F	ST or F	00	ST	Bc	#0
VN or N	VF	VF	0–PI	VF	all	F	00	F	F	0#
VF	VN	VF	N	VF	ST	Bc	00	F	F	11
VF	VF	VN	0–PI	F	all	ST or Bc	00	Bc	ST	11
F or VF	VF	VN	0	F	FF or Bc	ST or Bc	00	F	F	0#

how the activation between groups is produced. In this case, the genetic learning process does not demand a definition of activation among groups. In one RTCS, six different groups have been learnt, with the values of IT in condition part shown in Table V.

Each group contains a different number of rules and shares some condition values that reflect similar situations. In Tables VI–XI the different groups appear collected and the symbolic values for the part of condition can be observed, represented by the concepts s1, s2, s3, A (angle), d (distance), v1 and v2 (left and right wheels velocity values), followed by IT values in the condition part. Below

Table VII. Group 2 rules

Condition							Message			
s1	s2	s3	A	d	v1	v2	IT	v1	v2	IT
VN	VN	VF	0–PI	N or VF	FF	FF	01	ST	Bc	#1
N or VF	VN or N	F or VF	0 or 0–PI	F	F or FF	FF	01	ST	Bc	##
N or VF	VN	VF	0 or PI–2PI	F	FF	FF or Bc	01	FF	FF	#1
VN	VN or F	VN	0–PI	VF	F or FF	FF or Bc	01	Bc	ST	11
N	F	VN	0–PI	VF	Bc	ST	01	Bc	ST	#1
VN or N	N or VF	VN or F	0	VF	Bc	all	01	Bc	ST	##
N or VF	VN or N	VF	0–PI	VF	F	F	01	ST	Bc	0#
VN or N	VN or F	VN or F	2PI	VF	ST	all	01	Bc	ST	##
N	F	VN or F	PI–2PI	VF	FF	FF	01	Bc	ST	##
VN or F	VF	F	PI–2PI	F or VF	Bc	ST	01	Bc	ST	11
N	N or VF	VN	2PI or PI–2PI	F or VF	Bc	ST	01	FF	FF	##
F	VN or N	VF	PI–2PI	F or VF	F or FF	F	01	F	F	#1
VN or N	VN or F	VF	0–PI	F	ST or F	F	01	FF	FF	0#
VN or F	F	N	2PI or PI–2PI	VF	ST or F	Bc	01	Bc	ST	0#
VN	F	VN	0–PI	VF	ST	Bc	01	Bc	ST	##
VN	N	N	all	VF	ST or F	Bc	01	ST	Bc	##
VN	VN or N	VN or N	0–PI	VF	ST or Bc	Bc	01	ST	Bc	11
VN or N	F	VF	2PI or 0–PI	VN or F	FF	FF or Bc	01	ST	Bc	#1
F	VN	N or VF	PI–2PI	VF	FF	FF	01	ST	Bc	#1
N or VF	F or VF	VN or F	2PI	VF	all	ST	01	F	F	##
N	VN or N	VF	0 or 0–PI	VF	FF	FF	01	ST	Bc	11
N or VF	VF	VN	PI–2PI	N or VF	FF	FF	01	Bc	ST	##
N	N or VF	VN	2PI or PI–2PI	F or VF	Bc	ST	01	Bc	ST	0#
VN	VN or N	VF	0 or 0–PI	VF	F	F	01	ST	Bc	0#
N or VF	VN	VF	PI–2PI	VF	F	ST or F	01	ST	Bc	##
N	VN	VF	0	VF	ST or Bc	ST or Bc	01	ST	Bc	11
N or VF	VN	VF	PI–2PI	VF	FF or Bc	FF	01	ST	Bc	##
F	VN or F	VF	0 or 0–PI	N	ST or F	all	01	ST	Bc	0#
N or VF	VN	VF	PI–2PI	VF	F	ST or F	01	ST	Bc	#1

of condition values, message values: v1 and v2, and IT values in the message part are found.

Group 1 consists of 20 rules of the 119 that the learned RTCS contains. Analyzing the sensors values s1, s2 and s3 in the rules of group 1, this group represents situations of collision danger. The set of rules of group 1 are a part of “collision danger” behavior, defined by rules which are fired only when one sensor (S1, S2 or S3) is Very Near (VN). Besides, angle values (respect to the goal) compel the robot

Table VIII. Group 3 rules

Condition							Message			
s1	s2	s3	A	d	v1	v2	IT	v1	v2	IT
VF	VF	VF	0–PI	VN or F	all	ST or F	11	F	F	1#
VF	F or VF	F or VF	0 or PI–2PI	VF	ST	Bc	11	Bc	ST	#1
F or VF	VF	VF	0	VF	ST	Bc	11	FF	FF	#1
VF	VF	VF	2PI or 0–PI	all	FF or Bc	ST or Bc	11	ST	Bc	##
F or VF	VF	F or VF	0–PI	VF	FF	FF	11	FF	FF	11
F or VF	F or VF	VF	2PI	N	FF	FF or Bc	11	FF	FF	##
VF	VF	F or VF	2PI	N	F	ST or F	11	FF	FF	##
VF	F or VF	VF	0	VF	F	F	11	F	F	#1
VF	VF	VF	0	F	F	F	11	F	F	1#
VF	VF	VF	0	F	Bc	ST	11	F	F	##
VF	VF	F or VF	2PI or 0–PI	VN or N	ST	FF or Bc	11	F	F	##

to advance without turning when collision risk appears through sensors S2 or S3. In fact, analyzing the message part (velocities decided for each situation), turning and advanced of the robot are observed, but many rules (10 of 20 rules) force a straight trajectory although conditions represent situations of collision danger.

Group 2 contains 29 rules of the 119 ones that form the RTCS, so it has 50% more rules than in the previous group. In this group, the most important observed characteristic is that it contains many values “near” or “very near” in sensors s1, s2 or s3, which represent situations of collision danger, but is more general than in the previous one (group 1). The superset composed of rules of both groups (one and two) are entrusted with avoiding obstacles. In this case, the values of the decided velocities make the robot to turn, in order to avoid obstacles on the right/left side.

Group 3 consists of 11 rules of the 119 that the RTCS contains. In this group, there are less rules than in previous ones. Represented situations, in contrast to the two previous groups, do not contain any collision situation. In fact the rules of this group are related with angle values. Analyzing angle values for all rules, the robot seems quite aligned with the objective. As a result of the inference of each rule, messages sent to the robot compel it to advance in a straight trajectory, that corresponds to the situation where the robot is located forming a 0 or 2PI angle with the objective. Distance values to the objective do not seem, to be determinant to take decisions.

Group 4 consists of 21 of 119 rules that the RTCS contains. In this group, no clear tendency appears with respect to the general behaviors: “follow the objective” or “avoid obstacles”. Analyzing the rules, some of the proximity sensors s1, s2 and s3 has values of near, but this value appears in rules with Far or Very Far values, so that is no case of danger situations. Attending to the angle and distance values it is

Table IX. Group 4 rules

Condition							Message			
s1	s2	s3	a	d	v1	v2	IT	v1	v2	IT
N	N or VF	VF	0	VN or N	ST or Bc	Bc	#1	FF	FF	#1
VF	N	VF	2PI or PI–2PI	VF	Bc	ST	#1	F	F	#1
VF	N	N or VF	PI–2PI	VF	F or FF	F	#1	F	F	00
F	VF	N	0–PI	VF	Bc	ST	#1	Bc	ST	#1
N	VF	F	0–PI	VF	FF or Bc	ST	#1	Bc	ST	1#
N	N or VF	N	0 or 0–PI	all	Bc	ST or F	#1	Bc	ST	0#
VF	VF	N	all	VF	ST	ST or Bc	#1	FF	FF	#1
N	VF	all	2PI	VF	F	all	#1	FF	FF	1#
N or VF	N	VF	2PI or 0–PI	VF	FF	FF	#1	FF	FF	0#
F or VF	N	N or VF	2PI or PI–2PI	VF	ST or Bc	FF or Bc	#1	F	F	#1
F	N	VF	0–PI	F	F	F	#1	F	F	0#
F	N	VF	2PI or 0–PI	VF	all	F	#1	ST	Bc	#1
F	N	F or VF	0–PI	VF	FF	FF or Bc	#1	ST	Bc	00
N or VF	N	VF	0–PI	VF	FF	FF	#1	FF	FF	#1
F	F or VF	N	0 or PI–2PI	N or VF	FF or Bc	ST	#1	F	F	#1
All	VF	N	2PI or PI–2PI	N or VF	ST or F	Bc	#1	Bc	ST	1#
F	N	VF	2PI	VF	FF	F or FF	#1	ST	Bc	0#
VF	N	VF	0	VF	F	F	#1	F	F	#1
F	VF	N	PI–2PI	VF	FF	FF	#1	Bc	ST	00
F or VF	VF	N	0 or PI–2PI	VN or F	F	F	#1	F	F	#1
VF	F or VF	N	0–PI	F	F	F	#1	F	F	1#

observed that the robot, in almost all rules, is far from the objective and in general not aligned with it. These circumstances cause that rules compel the robot to turn toward the objective and to advance so that, thereinafter, some danger situation will be produced that groups 1 and 2 could resolve.

Group 5 is composed of 24 rules of 119 ones that form RTCS. This group is similar to the previous group with respect to the values of s1, s2 and s3, but opposite respect to angle and distance values. In this case, angle values define, in almost all rules, situations where the robot is aligned with the objective. As distance value, in most of the rules, corresponds with Far or Very Far distance situation, the combined effect of angle and distance values cause that rules compel the robot to advance straight to the objective.

Group 6 consists of 14 rules of 119 that RTCS contains. This group is similar to group 3 but the represented angle situations require rules that make the robot turn. So, groups 3 and 6 seem to be responsible for approaching the robot to the objective, while groups 1 and 2 seem to be responsible for avoiding obstacles.

Table X. Group 5 rules

Condition							Message			
s1	s2	s3	a	d	v1	v2	IT	v1	v2	IT
VF	N or VF	VF	all	L	ST or Bc	ST or Bc	10	FF	FF	10
VF	all	VF	0	all	FF or Bc	ST	10	FF	FF	10
VF	all	VF	0 or 0–PI	N	ST or F	F	10	F	F	10
VF	all	VF	0	all	FF or Bc	FF	10	FF	FF	0#
VF	N or VF	F	0–PI	VF	ST or Bc	Bc	10	F	F	10
VF	N or VF	F	0–PI	VF	all	F or FF	10	ST	Bc	#1
N or VF	F or VF	F	0–PI	VF	ST	Bc	10	Bc	ST	0#
VF	VF	N or VF	0–PI	N or VF	ST	Bc	10	FF	FF	10
VF	all	all	PI–2PI	VF	ST	ST or F	10	Bc	ST	10
VF	VF	N or VF	2PI or 0–PI	VF	F	F	10	F	F	#1
VF	VF	N or VF	PI–2PI	VN or N	FF	FF or Bc	10	FF	FF	0#
F or VF	VF	N or VF	2PI	VF	F	F	10	F	F	0#
N or VF	F	F or VF	2PI	VF	F	F or FF	10	F	F	10
VF	N or VF	N or VF	2PI	F or VF	F	ST or F	10	F	F	10
N or VF	F	all	2PI	L	ST or F	ST or F	10	F	F	#1
VF	VF	N or VF	2PI	L	FF	FF	10	FF	FF	##
VF	VF	N or VF	2PI	L	F	F	10	FF	FF	10
VF	all	VF	0–PI	L	F	F	10	F	F	#1
N or VF	N or VF	F or VF	0 or 0–PI	L	F	F	10	FF	FF	10
VF	F or VF	N or VF	2PI	VF	FF	FF	10	FF	FF	10
VF	VF	N or VF	2PI or PI–2PI	L	ST or Bc	ST	10	Bc	ST	##
VF	F or VF	N or VF	2PI or PI–2PI	L	ST	Bc	10	FF	FF	##
N or VF	VF	VF	0	N	FF or Bc	ST	10	F	F	10
VF	F	N or VF	0–PI	N	ST	FF or Bc	10	ST	Bc	10

6. Conclusions

The main goal of this work has been the development of a genetic learning system, where the solution of the problem is defined through a set of rules, each rule being a part of the solution. These problems can be approached through classifier systems, but two disadvantages appear: execution time and complex strategies generation. These two problems are especially relevant in robotics, where the rule systems are applied with relative efficiency and where, traditionally, learning systems have been applied.

This goal has been pursued from two different perspectives: on the one hand, the need of finding a CS that could work with limitations in execution time, and

Table XI. Group 6 rules

Condition							Message			
s1	s2	s3	a	d	v1	v2	IT	v1	v2	IT
FF	F	FF	2PI	VN or N	F	F or FF	1#	F	F	##
FF	F or VF	F	0	FF	F	F	1#	F	F	##
F	F	F or VF	2PI	N	ST or F	all	1#	ST	Bc	1#
FF	FF	F	PI–2PI	FF	FF or Bc	ST or F	1#	Bc	ST	11
FF	F	FF	0–PI	FF	FF or Bc	FF	1#	ST	Bc	##
F	FF	FF	0–PI	N or FF	Bc	ST	1#	ST	Bc	##
FF	F	F	2PI or 0–PI	F or VF	ST or F	all	1#	F	F	0#
FF	F	FF	0–PI	FF	ST or Bc	ST or Bc	1#	F	F	##
F or VF	F	F or VF	0	F or VF	FF	FF or Bc	1#	ST	Bc	00
FF	FF	F	0 or PI–2PI	FF	FF	FF	1#	ST	Bc	##
FF	F or VF	F	PI–2PI	FF	F	ST or F	1#	F	F	11
F	FF	F or VF	2PI	N	Bc	ST or Bc	1#	Bc	ST	##
F	FF	FF	PI–2PI	F or VF	Bc	ST	1#	Bc	ST	1#
FF	F	FF	0	F or VF	F	F	1#	F	F	##

on the other hand, the search for a solution to maintain the diversity in rules to elaborate complex strategies.

A rapid response is a common requirement for any learning system, especially when it works with temporary limitations. When classifier systems are applied in a changing environment, the temporary lag in the taking of decisions affects, on the one hand, the suitability of the output and, on the other hand, the output reward received from the environment. An unfitted reward produces a destructive effect on CS, since the reward is used to obtain the rules strength. The learning process is related with two levels (credits assignment and discovery); its performance is based on the rule strength, and such strength is the reward transposed to all rules through economical calculations among rules.

In order to develop a classifier system able to learn reactions and complex strategies to survive in a dynamic world, the information originated from the environment that CS receive will be referred to the previous CS situation. Therefore, the environmental information is updated on the world state that surrounds the system in each instant, and is injected at different instants (at the time when CS takes intermediate decisions). The output will not be only the action associated with the comparison of one rule, because solutions or intermediate decisions that contribute to the global solution are also considered. In this way, an adequate sequence of these intermediate solutions, considering actual environmental information, gives the global solution.

This work has been centered at the development of a CS named Reactive with Tags Classifier System (RTCS), which learns in a dynamic environment with temporary restrictions in execution. The system contains a set of mechanisms that allow the incorporation of new environmental information in the process of taking decisions. This process allows rule sequence (chaining rules at different execution instants) and break of the sequence to provide a reactive output. Thus, the developed RTCS has the capacity of learning reactions and strategies, so the dilemma between reactive and planned systems could be surpassed. A classifier system that operates this way allows to solve problems independently of the environment in which they are developed, with no previous knowledge of the problem, through the credit assignment and genetic algorithms.

To evaluate the RTCS, a problem in a dynamic environment which required a reactive behavior has been chosen. An environment of these characteristics appears in robotics, particularly in navigation of a robot, moving in a world with obstacles where the robots goal is to reach a predefined point. This problem, from the learning point of view, is considered to be complex enough if the decision must be obtained in real time, since the environment continues to change during the time of taking a decision, or, from another point of view, since the robot is moving while the decision is taken.

The analysis of groups shows the capacity of the RTCS to evolve coherent rule groups. This coherence can be seen from two points of view: on the one hand, all rules of each group represent similar situations and give similar outputs and, on the other hand, independent behaviors are discriminated, so groups are quite independent. Furthermore, only groups necessary to solve the problem are generated, not all the possible groups. This allows to conclude that the number of bits to represent possible groups must permit the evolution of all necessary groups and the RTCS will learn the number of groups that are actually necessary.

References

1. Booker, L. Goldberg, D. E., and Holland, J. H.: Classifier systems and genetic algorithms, *Artificial Intelligence* **40** (1989), 235–282.
2. Brooks, R. A.: Intelligence without representation, *Artificial Intelligence* **47** (1991), 139–159.
3. Dorigo, M.: ALECSYS and the autonoMouse: Learning to control a real robot by distributed classifier systems, *Machine Learning* **19** (1995), 209–240.
4. Golberg, D. E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
5. González, A. J. and Dankel, D. D.: *The Engineering of Knowledge-Based Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
6. Grefenstette, J. J.: Credit assignment in rule discovery systems based on genetic algorithms, *Machine Learning* **3** (1988), 225–245.
7. Holland, J.: *Adaptation in Natural an Artificial Systems*, Univ. of Michigan Press, Ann Arbor, 1975.
8. Holland, J.: Adaptive algorithms for discovering and using general patterns in growing knowledge bases, *Internat. J. Policy Analysis Inform. Systems* **4** (1980), 245–268.

9. Holland, J.: Properties of the Bucket Brigade, in: *Proc. of Internat. Conf. on Genetic Algorithms and Their Applications*, Vol. 1, 1985, pp. 1–7.
10. Holland, J.: A mathematical framework for studying learning in classifier systems, *Physica D* **22** (1986), 307–317.
11. Holland, J.: Properties of the Bucket Brigade, in: J. J. Grefenstette (ed.), *Proc. of an Internat. Conf. on Genetic Algorithms and Their Applications*, 1986.
12. Holland, J. H.: *Hidden Order: How Adaptation Builds Complexity*, Addison-Wesley, Reading, MA, 1995.
13. Isasi, P., Berlanga, A., Molina, J. M., and Sanchis, A.: Robot controller against environment, a competitive evolution, in: *15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics*, Special Session on Evolution Computation, Germany, 1997.
14. Lee, M. A. and Takagi, H.: Integrating design stages on fuzzy systems using genetic algorithms, in: *Second Internat. Conf. on Fuzzy Systems*, 1993, pp. 612–617.
15. Matellán, V., Fernández, C., and Molina, J. M.: Genetic learning of fuzzy reactive controllers, *Robotics Autonom. Systems* **25**(1/2) (1998), 33–41.
16. Matellán, V., Molina, J. M., Sanz, J., and Fernández, C.: Learning fuzzy reactive behaviors in autonomous robots, in: *Proc. of the 4th European Workshop on Learning Robots*, Germany, 1995.
17. McKerrow, P. J.: *Introduction to Robotics*, Addison-Wesley, Reading, MA, 1991.
18. Mitchell, M.: *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.
19. Molina, J. M., Sanchis, A., Berlanga, A., and Isasi-Viñuela, P.: Evolving connection weight between sensors and actuators in robots, in: *IEEE Internat. Symposium on Industrial Electronics*, 1997.
20. Mondada, F. M. and Franzi, P.: Mobile robot miniaturization: A tool for investigation in control algorithms, in: *Proc. of the 2nd Internat. Conf. on Fuzzy Systems*, San Francisco, USA, 1993.
21. Sanchis, A., Molina, J. M., and Isasi, P.: Classifier systems for learning reactions in robotic systems, in: *The 1st Internat. Workshop on Machine Learning, Forecasting and Optimization (MALFO'96)*, 1996, pp. 153–159.
22. Sanchis, A., Molina, J. M., and Isasi, P.: Learning reactive behavior for autonomous robots using classifier systems, in: F. L. Silva, J. C. Principe, and L. B. Almeida (eds), *Spatiotemporal Models in Biological and Artificial Systems. Frontiers in Artificial Intelligence and Applications*, Vol. 37, IOS Press, 1997, pp. 152–159.
23. Shu, L. and Schaeffer, J.: HCS: Adding Hierarchies to classifier systems, in: *4th Int. Conf. on Genetic Algorithms*, 1991, pp. 339–345.
24. Sommaruga, L., Merino, I., Matellán, V., and Molina, J.: A distributed simulator for intelligent autonomous robots, in: *Fourth Internat. Symp. on Intelligent Robotic Systems – SIRS'96*, Lisboa, Portugal, 1996.
25. Weiß, G.: Hierarchical chunking in classifier systems, in: *Proc. of the 12th Internat. Conf. on Artificial Intelligence*, 1994, pp. 1335–1340.
26. Wilson, S.: Knowledge growth in an artificial animal, in: *Proc. of the 1st Internat. Conf. on Genetic Algorithms and Their Applications*, 1985, pp. 16–23.