Reinforcement-Based Learning with Automatic Categorization

Josep M Porta

Institut de Robòtica i Informàtica Industrial (CSIC — UPC), Gran Capità 2-4, 08034, Barcelona (SPAIN), jporta@iri.upc.es

Abstract. In this work, we present a reinforcement-based learning algorithm that includes the automatic categorization of both sensors and actions. The categorization process is prior to any application of reinforcement learning. If categories are not at the adequate abstraction level, the problem could be not learnable. The categorization process is usually done by the programmer and is not considered as part of the learning process. However, in complex tasks, environments, or agents, this manual process could become extremely difficult. To solve this inconvenience, we propose to include the categorization into the learning process. We sketch an algorithm to automatically learn to achieve a task through reinforcement learning that works without needing a previous categorization process. First results of the application of this algorithm are shown.

1 Introduction

Continuos improvements in areas such as mechanical engineering and micro-electronics have given us the possibility of building autonomous robots with increasingly sophisticated sensorial and motor systems. The control of this kind of robots to accomplish complex tasks in dynamic environments is one of the challenges of Artificial Intelligence but we are far from achieving it.

In the last years, some authors (Brooks, 1991) have argued that traditional approaches of artificial intelligence based on high level reasoning and planning are not adequate for the type of problems that arise when controlling a real robot in real time, and new architectures have been proposed. The use of these new architectures allowed to achieve complex tasks with controllers based on simple principles (Brooks, 1986), (Connell, 1989). But the programming of these controllers is not free of problems. Behind each success of these approaches there is usually a programmer that has spent lots of hours designing the controller, developing each one of its modules and adjusting their parameters and interactions until the desired behavior is obtained. If we want to address more complex tasks and environments, this programming process must be alleviated, and the application of machine learning techniques is a promising way to explore. The idea of automatic learning of robot controllers has been present in the new paradigms of robot control architectures from the very beginning (Dorigo and Colombetti, 1994), (Maes and Brooks, 1990), (Mahadevan and Connell, 1992), (Mataric, 1990).

A learning paradigm can be classified according to the amount of information that the designer directly provides to the learning agent[1]. At one end of this continuos spectrum (closer to the manual programming) we find the supervised learning approaches in which the designer informs the agent about which are the adequate actions (with respect to the current task) to be executed in some specific situations and the objective of the learning algorithm consists in generalizing the given information to find out the correct actions in all the possible situations. At the other end of the learning paradigms, there are the *reinforcement learning* systems (Kaebling et al., 1995). In these systems, the designer only informs the agent when the task has been successfully completed. Between the two extremes of the classification, there is

[1] There exits also unsupervised learning approaches in which no information is given to the learning agent but they are not task-purposive.

a variety of learning systems that the programmer can use depending on her knowledge of the task and on how easy is it to transfer this knowledge from the designer's point of view to the robot's point of view. In many problems concerning autonomous robots the reinforcement learning paradigm is the only adequate one.

Reinforcement learning has been extensively studied since the origins of artificial intelligence and even before. However, as it happened with the traditional artificial intelligence architectures, learning paradigms must be adapted to be successfully applied to autonomous robots. Since now, existent reinforcement learning algorithms have been mainly focused on how to find the correct links between perception and action without paying much attention on how perception and action must be pre-processed, so that relevant mappings between them can be established. In general, current reinforcement learning algorithms depart from the basis that there exists meaningful states for the solution of the task and adequate actions to go from one state to another (see (Sutton and Barto, 1998) page 61 for a good explanation on this assumption). The process of definition of these states and actions is called *categorization*. This process is in charge of the programmer and is often implicit in the application of the reinforcement learning algorithms. The categorization process can be the most difficult stage of the solution of a task (the proper interpretation of the sensor readings can transform an apparently complex task in a simple one) so, if it is done by the programmer, we could be systematically confronting our reinforcement learning algorithms with the easiest part of the problem.

In this work, we present a reinforcement based learning approach that derives adequate sensor and action categories for solving a task only from the interaction of the robot with the environment and the information given by the reinforcement signal. Our approach uses the so obtained categories to construct a controller that achieves the task.

In the next section we analyze in more detail the generalities of the existent reinforcement learning algorithms, and we show the necessity of a categorization process in both sensors and actions when these algorithms are applied to complex agents such as autonomous robots. In section 3, our reinforcement based learning approach is sketched, and in section 4 we describe the implementation of a reduced version of this system and shows some preliminary results. Finally, in section 5 we extract some conclusions of our work and outlines future ways in which our system should be extended to cope with more complex learning problems.

2 Reinforcement Learning and Categorization

The reinforcement learning paradigm has been extensively studied in areas such as animal ethology, automatic control, and more recently artificial intelligence. A general description of a reinforcement learning situation includes the following elements:

- A definition of the task: The only strictly needed information about the task is a signal, called the *reinforcement signal*, that becomes active when the task is achieved. This signal is. If more information about the task is available (as for instance, correct or incorrect actions in some situations, or necessary preconditions for achieving the task), it can be added to the reinforcement signal to help the learner.
- An environment in which to accomplish the task. Depending on the kind of environment (static, dynamic,...) the resolution of the task can be less or more difficult.
- An agent that must complete the task. This agent can execute actions. The solution of a task consists in determining adequate actions to activate the reinforcement signal as frequently as possible.

The first attempts to formalize the framework of reinforcement learning were at a high level of abstraction, and the resulting formalization was not as general as the previous description. The main assumption of this formalization is to consider the environment as a state machine controlled by the agent, so that the interaction between the environment and the agent is arranged in a two step loop: In the first step, the agent perceives the state of the environment and, in the second one, it performs an action that produces a change in this state. In this formalization we have:

- States: Representing relevant situations for the achieving of a task to which the agent should respond with the correct decision. The agent can directly perceive the state of the system, or at least obtain sensor readings that depend probabilistically (in a constant way) on the state.
- Actions: Devised as a list of options from which the agent picks the most appropriate for each state. At each moment the agent only performs one action from the list.
- Reinforcement function: It is a function that maps transition between states to a real numbers. In general this function is zero except for some transitions that are considered the goals of the current task (for instance, reaching a specific position, grasping an object,...). The reinforcement

function is defined from the point of view of the programmer, and it is the only way in which she transfers her knowledge about the task to the agent. The task is more easily learned if the agent has a set of states and actions equivalent to that of the programmer. In general it is assumed that the programmer uses the same set of states and actions that the agent but as we show in the next sections, in complex tasks this is not always true.

- Policy: A mapping from states to actions. Each policy is evaluated according to a criterion that depends on the reinforcement function. Reinforcement learning algorithms aim to find the optimal policy for the task at hand.
- Model of the agent-environment interaction: Including the transition probabilities from one state to another under the execution of each action.

This formalization was quickly accepted by the artificial intelligence researchers coming from areas such as automated reasoning or planning, because they also used to tackle problems at a high abstraction level, in which states and actions to move from one state to another were easy to define. So, the reinforcement learning algorithms used in artificial intelligence heavily rely on the assumption of the availability of high level states and corresponding actions that cause transitions between these states according to well defined probability distributions. As far as this assumption is not fulfilled, the algorithms do not work properly. In practice, the definition of states is often done taking the current sensor readings as the present state and the set of actions is usually defined as the whole set of the elementary actions executable by the robot. Unfortunately, this simple solution does not work properly in most autonomous robots applications. The reason is that sensor readings and elementary actions conform a too large search space (Utgoff and Cohen, 1998), and that, in a complex autonomous robot, it is almost impossible to consistently predict the next sensor readings after the execution of an elementary action. This is in contrast with the assumptions underlying the reinforcement learning algorithms explained above, so with this definition of states and actions, they can hardly accomplish even the simplest tasks when applied to autonomous robots.

A first attempt to solve these problems, consists in making clusters of sensor readings to define and picking only special combinations of actions as the set of actions. This reduces the spaces manipulated by the learning algorithm. Additionally, if the clusters are properly defined, the relation between the inputs and the actions manipulated by the learning algorithm becomes more predictable. For instance, in (Mahadevan and Connell, 1992) sensor readings are conveniently grouped prior to the definition of states both manually and through an automatic process (thus, infrared sensor readings are clustered in two classes using programmer defined criteria, so that only two situations for detected objects can be distinguished: near and far). In the same work, only five actions (forward, small turn left/right, and large turn left/right) are considered despite the fact that the robot could perform many other movements.

The process of selecting appropriate aggregations of sensor readings and choosing adequate combinations of elementary actions so that they can be used as a basis to understand and accomplish a task is called *categorization* (of sensors and actions) and its result are the *sensor* and *action categories*. The categorization of sensors and actions increases the level of abstraction up to a point in which the task at hand is achievable: if appropriate categories are determined by the programmer, the task of the learning algorithm will be considerably alleviated. However, for complex tasks and environments, the manual process of categorization could become extremely complex, and an automatic categorization process will become necessary. Additionally, even if a human programmer can do the categorization, an automatic mechanism may be helpful since it could discover sensor relations or action combinations not foreseen by the programmer (the so called *frame of reference problem* (Pfeifer, 1995)).

Sensor categorization can be seen as a process of interpretation of sensor readings so that objects and situations relevant for solving the problem are identified. By its side, action categorization involves identifying the dynamics of those objects and situations and specially their interaction with the robot. For instance, if we want a robot to fetch a soda can that is at the end of a corridor, then it is helpful if the robot can identify things such as 'corridor' or 'soda can' and it knows how to 'go along a corridor' or how to 'pick a can'. If these objects and actions are not available to the robot, then the task could be not learnable.

Usually, designers define a task at a high level of abstraction using its own concepts and actions. However, those high level concepts and actions can be recursively decomposed in lower level concepts and action into elementary ones (directly coupled to sensor readings or constructed from elementary actions). So categories are not monolithic entities but there is a hierarchy from low abstract categories to more elaborate ones. This characteristic suggests that it is sensible to affront the category formation process departing from simple problems in which simple categories are already needed and use these categories as a basis to confront more complex learning tasks. This incremental teaching process is already used in

reinforcement learning and is known as *shaping* (Dorigo and Colombetti, 1994).
As noted before, sensor categories must identify interesting objects or situations for the current task. In the absence of reinforcement, however, it is not possible to know whether or not a sensor category is interesting for the current task. What seems sensible to do is to identify objects and situations characteristic of the environment so that, when the reward becomes active, it can be accounted for in terms of these already identified objects which provide information at a higher level than the basic sensor readings. This process of discovering environment related categories can be based on the analysis of the environment regularities according to the activation frequency of different sensor combinations.
It is important to remark that typical reinforcement learning algorithms (such as Q-learning (Watkins and Dayan, 1992)) do not take advantage of possible environment regularities: they are designed to learn policies for any arbitrary problem as efficiently as possible. In our case, we prefer to favor the discovering of environment regularities since, in our opinion, this can help the learning process in cases in which the reinforcement signal is activated with low frequency.
Once environment related sensor categories are found, we can begin to identify interesting action categories. The idea is that those action categories that makes sense to execute are those that have a predictable effect on the available set of sensor categories. So examining the correlation between the execution of an action category and the activation or deactivation of sensor categories, relevant action categories to interact with the environment can be determined.
The advantage of finding task independent sensor and action categories is that they can be useful for many similar tasks and environments.
The problem of the task independent categories is that there are too many of them since there exists lots of possibly interesting objects in the environment to be identified and manipulated. So the information provided by the reinforcement signal must be used to filter them. What we have to do is to discover which sensor and action categories are correlated in any way with the reinforcement signal. Those reinforcement related categories can be used to synthesize an adequate controller for achieving the task at hand.

3 An Automatic Categorization Algorithm

In the previous section we have analyzed the role of categorization in reinforcement learning and we have outlined a way by which adequate categories can be obtained by observing the environment, from interacting with it, and using the information given by the reinforcement signal.
In this section we adopt a more practical point of view and we introduce specific structures to combine sensor readings into sensor categories and elementary actions into action categories. We also explain how the processes for category selection can be applied on these structures. Additionally we indicate how the resulting sensor and action categories can be combined in *behaviors* that conform a controller for accomplishing the task at hand.

3.1 Sensor Categorization

A simple sensor category can be devised as a combination of sensor readings. From the different possible ways to define groups of sensor readings (clustering is one of the most typical) we choose a general method that consists in defining logical predicates over the sensors. The idea is to define functions over readings coming from different sensors to produce a binary output. For instance, the category "*corridor*" can be defined based on the readings of the distance sensors of the robot as:

$$(\textit{left distance} < \textit{THRESHOLD})\ \textbf{\textit{AND}}$$
$$(\textit{right distance} < \textit{THRESHOLD})$$

In this example, the sensor category uses two sensors (left and right distance sensors), and the '<' and 'AND' operations are used to produce the desired binary output.
This sensor aggregation mechanism is very general since almost all kinds of categorization performed up to this moment, either by the programmer or automatically, fit in this framework (the differences between the existent approaches are the functions used to combine the sensors). If some knowledge is available about the environment or the task, the user can reduce the generality of the system by selecting those sensors that can be combined or restricting the set of applicable operations. This will alleviate the work of the automatic categorization.
It is interesting to note that, in general, sensor categories are not equivalent to state identifiers. A classical reinforcement learning state (as described in section 2) encompasses all the information available at any given time, and a unique action to be performed by the robot is determined from it. In contrast, sensor categories are intended to identify objects or situations in the environment, and can be composed in many ways to determine part of the actions of the robot. Lots of sensor categories can be active simultaneously denoting interpretations of different subsets of the

perception of the robot, or even of the same subset of the perception but with alternative criteria or goals. It is part of the task of an automatic categorization mechanism to determine which active categories are more relevant.
Each syntactically correct binary function over the set of sensors is a candidate to a sensor category potentially useful for the current task. We set up a genetic algorithm (Goldberg, 1989) to search in the space of sensor categories applying the usual genetic operators (crossover and mutation). The fitness function is defined using the three criteria for identifying adequate categories explained in the previous section:

- Correlation with regularities of the environment: This basically consists in observing the activation frequency of each function. Those with too low or too high activation frequency are rejected.
- Correlation with the robot-environment interaction: Those functions that respond in a predictable way to the execution of an action category are preferred because they are capturing part of the dynamics of the interaction of the robot with the environment.
- Correlation with the reinforcement signal: The functions whose activation or deactivation is strongly correlated with the activation of the reinforcement signal are prioritized.

3.2 Action Categorization

We devise an action category as a special combination of commands sent to motors. For instance, in a wheeled robot with two independent motors (for left and right wheels), one could define action categories such as:

Advance:
(left motor forward at full speed)
AND SIMULTANEOUSLY
(right motor forward at full speed)

Turn left:
(left motor forward at half speed)
AND SIMULTANEOUSLY
(right motor forward at full speed)

The range of definable action categories depends on the kind of orders allowed to be sent to motors (constant values, values parameterized as functions of the sensor readings or previous motor orders,...) and on the available combinations of elementary motor commands (SIMULTANEOUSLY, SEQUENCE,...). The range of these two degrees of freedom of the action categorization system must be set by the programmer: if she sets them in a too general way then the automatic categorization mechanism will work slowly and if she sets them in a too limited way then the automatic categorization would fail.
As in the case of sensor categories, an action category has not to be seen as an action in the traditional sense of the reinforcement learning algorithms, in which only one action is executed at a time. In our approach many action categories can be executed at the same time. This is a characteristic that is already present in the classifier systems paradigm (Goldberg, 1989). Even more, a single series of low level actions can be interpreted as the execution of different action categories. For instance, moving along a corridor can be interpreted as "going to an intersection", "going to the door", "escaping from a maze",... The correct interpretation will depend on the task to be performed or on the level of abstraction at which the task is interpreted.
The space of syntactically possible action categories can also be searched using genetic algorithms. In this case, the fitness function is similar to that of sensor categories but is based on only two criteria (correlation with the robot-environment interaction and correlation with the reinforcement signal) since it would make little sense to base it on the regularities of the actions of the robot.

3.3 Behaviors

In order to accomplish the task at hand, sensor and action categories must be related. A classical reinforcement learning algorithm can not be used since the set of categories changes with time (and those algorithms assume predefined state and action sets) and because, as explained before, sensor and action categories are not equivalent to classical reinforcement learning states and actions.
In our schema, the relation between sensor and action categories is achieved through the creation of *behaviors*[2]. These structures define acting rules with the following elements:

1. A combination of sensor categories acting as the condition part of the rule.
2. An action category to be executed when the condition holds.
3. The expected effect of the execution of the action category on the sensor categories. This effect can be the activation, deactivation or no alteration of a sensor category. This is used to check the correct execution of the behavior when it is actually performed.

[2] We call them behaviors because of their similarity with those used in the behavior based architectures (Brooks, 1991).

A behavior produces a pattern of interaction of the robot with the environment that is more in this environment than random actions. Behaviors are created from the most correlated sensor and action categories resulting from the genetic searches described in previous sections. In exploration mode, the robot is allowed to execute random elementary actions, and the sequences of sensor readings and actions actually executed are interpreted using the sensor and action categories and their correlation is computed. This kind of analysis permits a high degree of parallelism since the same sequence of sensor readings and actions can be interpreted in different ways giving rise to a parallel statistic actualization. Once enough evidence is collected about the effects of one action category over a sensor category in a specific context, the corresponding behavior is generated. From this moment, the robot includes this behavior in its action repertory and executes it when appropriate. In this way, the robot progressively drifts its actuation from a random walk to more adequate ways to interact with the environment. Behaviors using reinforcement related categories are prioritized, resulting in the eventual learning of the task.

A behavior can be evaluated according to its relation with the reinforcement signal. Our simple credit assignment mechanism works as follows. The utility of each behavior is adjusted according to the magnitude of the obtained reinforcement signal and the time difference between the execution of each behavior and the moment of the reward activation. The behavior utilities are used for two purposes. First, they are used to select which behavior to when sensor readings allow incompatible behaviors to become active. In second place, the behavior utility is used to eliminate useless behaviors.

Once an interesting behavior is found, two things are done:

- The firing conditions of the behavior are taken as a subgoals to be achieved through other behaviors. This reward transmission system allows the construction of chains of behaviors including elements not directly correlated with the reinforcement but useful to reach it.
- The sensor and action categories used in a interesting behavior can be considered categories with a special meaning in the current environment. Those categories are adequate candidates to be used as inputs to construct more abstract categories.

In figure 1, you can see a schematic representation of the complete automatic categorization system described in this section.

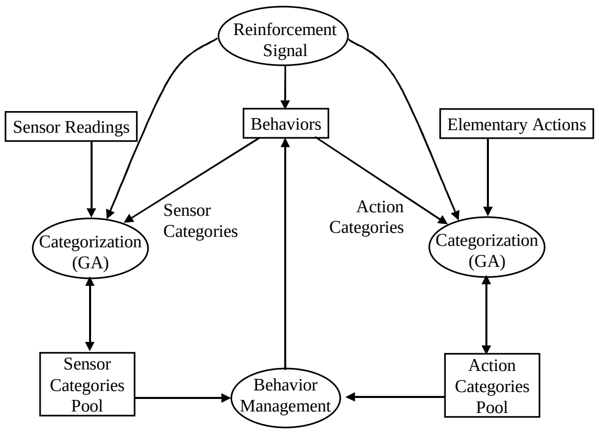


Figure 1: Schematic representation of the automatic categorization system. Boxes represent structures while circled texts stand for processes to be performed on these structures.

4 Implementation and Examples

We have implemented an almost complete version of the automatic categorization mechanism described in the previous section (the automatic creation of high level categories departing form those used in behaviors is not implemented yet).

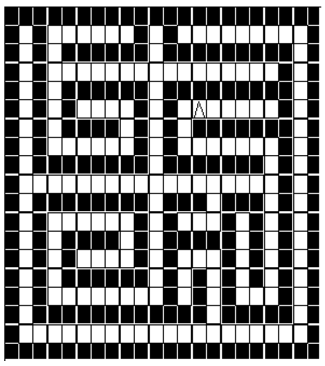


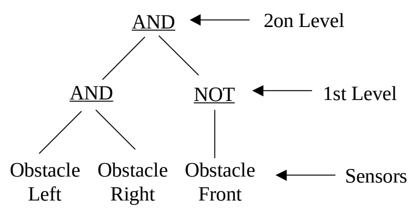
Figure 2: Simulation environment. The '$\wedge$' is the agent that moves in this world (advancing, turning right or turning left.

We have performed the first tests of our categorization system in grid world environments (a classical type of environments in reinforcement learning). This kind of environments are not the most appropriate to test our system since it has been designed for real robots tasks, but we have chosen them because they are easy to be simulated and because it simplicity allow us to concentrate on aspects such as category and behavior generation or reward transmission without being overwhelmed by aspects derived from the environment complexity. In figure 2, we show the aspect of our simulation environment.

Every one of the eight sensors of our simulated robot (denoted as Si) returns a binary value indicating whether or not there is an obstacle in one of the positions that surround the robot. These positions are enumerated with respect to the robot position and orientation as follows:

7	0	1
6	^	2
5	4	3

The sensor categories are defined as two-level tree expressions with operations over the sensors. The functions used to construct these boolean expressions are AND, OR, NOT and IDENTITY. With these restrictions, we can form sensor categories such as



that indicates the situations in which the robot is in a corridor and nothing blocks its way.
As far as the motor apparatus is concerned, we are working with a robot with only one motor that can execute three different actions: move forward, turn left, and turn right. These are the elementary actions used to construct action categories. Since it is not possible to execute elementary actions in parallel, the action categories we form consist in sequences of elementary actions.
The behaviors generated by this reduced version of our categorization algorithm use only one sensor category as a triggering condition and we only create two types of behaviors (IF and WHILE behaviors, with the usual meaning of these constructions).

4.1 Categorization

Sensor categories may arise either from the analysis of environment regularities or from the information given by the reinforcement signal. From the point of view of the designer, a grid world can be decomposed in some basic elements such as walls, intersections, corridors,... The agent executing our algorithm also detects basic objects that can be used to interpret the environment but it must be realized that the agent always perceives the world from its own point of view so, sometimes, the objects it identifies or the ways in which it identifies them can be very different from what an external observer could expect. For instance, in an empty world in which the only observable objects are the walls that surround the working area of the agent, the designer could define simple wall detectors as for instance:

Wall position with respect to the robot	Function that represents the sensor category
FRONT	S0
RIGHT	S2
BACK	S4
LEFT	S6

Our agent, a part from these definitions, can find other equivalent ways to identify walls that are not as evident as these ones. For instance, a wall in front of the robot can be represented by a functions as:

S1 AND S7

In other cases, the agent constructs sensor categories as

S0 OR S1

that detects wall either in front and at the right of the robot. This category, that is not naturally identified by the designer, could be useful for some task. Its potential utility for the task at hand will be assessed from the information given by the reinforcement signal.
This disagreement between the categories that the designer would generate and the ones actually generated by our systems is more evident in complicated environments.
In what concerns action categories, we construct them as sequences of elementary actions. For instance, our system could generate an action category like

```
SEQUENCE
        Turn Left
        Advance
```

that could be useful, for instance, for entering in a corridor at left of the robot.

4.2 Behavior Creation

Behaviors are created according to the correlation between sensor and actions categories making special attention to those categories related with the reinforcement signal. In the empty of the previous section, our system generates behaviors like:

WHILE S3 DO Advance

that can be used in many cases to follow a wall or

WHILE NOT(S1) DO Advance

that takes the robot to the wall in front of it.

Many other behaviors not so evidently (from the point of view of the programmer) useful for any task are also created.
The role of those behaviors created as a result of the environment regularities is more evident in tasks with sparsely activated reinforcement signals. This is the case of the environment of figure 1 when the task of the robot is to reach a feeding point (that we put at the blocked end of each corridor) as fast as possible. This task can be accomplished combining two types of behaviors. The first type of behaviors should take the robot to the end of a corridor and include elements as for instance:

WHILE NOT(S0) DO Advance

and the second type of behaviors should produce turnings at intersections:

IF NOT(S2) DO Turn Right

(in this environment turning right is more relevant than turning left because the main part of the corridors form right oriented spirals).
These two kinds of behaviors are discovered attending only to the environment regularities (that are quite evident). The information of the reinforcement signal (that is difficult to activate when performing a random walk) is only used to adjust the utility value of these behaviors.
Our algorithm, after visiting 100 reward situations, produces a controller that, in average, takes the robot to the blocked end of a corridor in less than 350 steps, while a random walk takes in average more than 800 steps in finding a feeding position. In many cases, the resulting controller is far more efficient than average and needs only around 30 steps to reach a goal situation.
Despite the controller can contain only behaviors based on environment regularities, in general, reinforcement based behaviors are more useful.

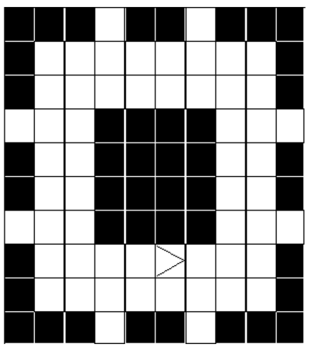
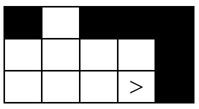


Figure 2: A room with eight exits.

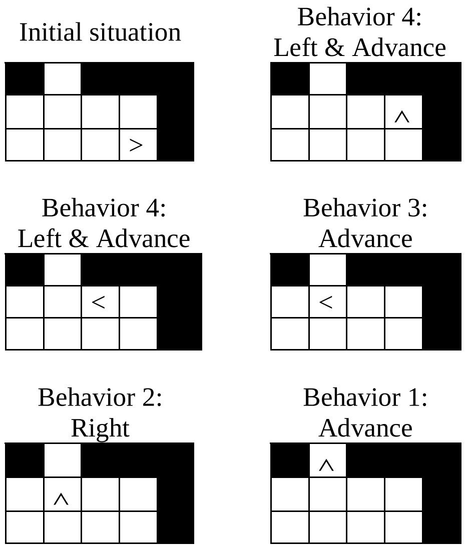
Consider for instance, the environment of figure 2 than can be seen as representing a room with eight exits. The goal of the robot is to reach an exit to go out of the room. The final controller for this task can include (among others) the following behaviors[3]:

```
1.- IF S1 AND S7 AND NOT(S0) DO
        Advance
2.- IF S1 AND S3 AND NOT(S2) DO
        Turn Right
3.- WHILE S2 AND NOT(S7) DO
        Advance
4.- WHILE (SO AND S7) OR (S1 AND S3) DO
        IN SEQUENCE
            Turn Left, Advance
```

Behavior 1 is directly related to the reinforcement signal. Observe that behavior number 4 includes two actions that must be executed in sequence. This two consecutive actions constitute an action category that has been found to be useful for the current task. Using these four behaviors and departing for a situation like:



the controller executes the following sequence of behaviors:



so the exit of the room is found. Observe that the behaviors conform an adequate sequence in the sense that one produces the activation of the following until the goal is achieved. This is a result of the automatic subgoal generation performed by the algorithm: the firing conditions of behaviors that carry to a reward situation (behavior 1 in this case) are considered as a subgoal situation to be achieved and new behaviors for reaching it are generated (behavior 2 in the example). The iteration of this process produces a chain of

[3] These behaviors are extracted from a controller produced by our algorithm. For clarity of the presentation only the most interesting behaviors are shown.

consecutive behaviors as that shown in the example. It is interesting to note that, when behavior 2 is executed, behavior 4 could also be executed but our action selection mechanism (based in the utility of each behavior) decides to execute behavior 2 since it is directly correlated with behavior 1 (that moves the robot to a reward situation in only one step) while behavior 4 is not.

5 Conclusions and Future Work

In this work we have analyzed a process called categorization that is previous to any application of existent reinforcement learning algorithms. This process is usually responsibility of the programmer and not considered as part of the learning process. The categorization increases the abstraction level of the inputs and outputs manipulated by the learning algorithm up to a point at which the task is learnable. The manual identification of sensor and action categories is possible only in simple environments, tasks and robots, but if we want to solve complex problems in dynamic environments using agents with complicated sensor and motor systems (as is the case of autonomous robots applications) we have to devise automatic forms to create sensor and action categories.
With this objective in mind, we have analyzed the role of categorization in reinforcement learning and we have proposed a method to accomplish a task through reinforcement learning without assuming a previous manual categorization process. This algorithm generates sensor categories, action categories and behaviors that relate the previous elements. After a period of exploration of the environment, a set of behaviors is generated forming the controller of the robot and containing what is traditionally called a *policy* in reinforcement learning (a mapping from sensors to actions that accomplishes the task). Our system is related with many other disciplines:

- Classifier systems: These systems (Goldberg, 1989), (Dorigo and Colombetti, 1994) also find a set of rules to achieve a task using a reinforcement signal. The main differences of our work with respect to classifier systems are that we build sensor and action categories that are correlated to form behaviors while classifier systems concentrate on the behavior construction without taking into account the category formation. Additionally, classifier systems select behaviors using only the information given by the reinforcement signal and do not create behaviors attending only to environment regularities. Other differences between the two approaches are that we allow a more general set of operations to be applied over the inputs (classifier systems only use the AND operation) and that we define action categories, while classifier systems only use simple actions.
- Behavior based controllers: The controller learned by our algorithm is composed of behaviors as proposed in many robot control architectures (Brooks, 1991). Our action selection mechanism (a basic piece of the behavior-based approaches to robot control) is based on the utility of each behavior calculated from the reinforcement signal. Note that the resulting controller will be reactive only if the operations over the sensor inputs are reactive.
- Genetic programming: We learn a controller for a task using genetic algorithms. Other authors have used genetic algorithms to evolve programs that perform a task (Koza, 1992).
- Multitask learning: The algorithm can generate categories derived from environment regularities and not directly related with the task to be accomplished. This categories are potentially useful for many tasks in similar environments. This reuse of the experience is also present in an approach to machine learning called multitask learning (Caruana, 1993).

Taking into account the history of the reinforcement learning algorithms we can say that our system goes a step further than the usual reinforcement learning algorithms. The first algorithms to learn through reinforcement were developed in the process optimization area and assumed the knowledge of the complete model (states, actions and probability transitions between them) of the system to optimize. The next generation of reinforcement learning algorithms relaxed this hypothesis in the sense that they do not require the knowledge of the model of the process but as we have noted in this work, they heavily rely in the correct definition of states and actions. In some cases (as for instance in the classifier system paradigm), this hypothesis has been somehow relaxed: states are defined automatically but relevant categories to do that must be predefined. The algorithm we have proposed completely relaxes this assumption and can work in cases in which the definition of states and actions is not available at the beginning of the learning process so we will be able to solve problems not solvable up to now. This set of new attainable problems includes interesting cases such as those posed when trying to learn a complex task with an autonomous robot. The drawback of relaxing the hypotheses is that the learning process is in charge of solving more things than a usual reinforcement learning algorithm and so it is less efficient as far as convergence time is concerned (so using our approach makes sense when in those situations in which no other algorithm is applicable). However, what is remarkable of our approach is what we are trying to learn more than how efficiently we learn it.

Results obtained with our algorithm are very preliminary but promising. We are working in simulated, simple worlds but adequate concepts and actions to interact with these environments are discovered helping to achieve the proposed tasks. Next steps in the development of our systems passes through including the capacity of generating high level sensor and action categories. Another important aspect to analyze in more detail is the case of robots with more than one motor and with motors able to execute a wider range of elementary actions than the ones used in our examples. Another interesting capability to add to our system is that of the reutilization of the structures of category definitions so we can apply the structure of an adequate category on different but related sensors and actuators (this is the macro capacity that (Brooks, 1992) demands to learning algorithms when applied to the control of autonomous robots).

With these additions, our system will be almost complete and ready to be applied to an autonomous robot. Our final objective is that of learning an autonomous robot controller to accomplish a complex task in a real environment without any previous pre-process of neither sensors nor actions. This, as (Brooks, 1991b) said, is the most complex thing one can attack in the area of learning within the field of autonomous robots and probably we are still far from solving it. However, it is an objective that must faced if we want to accomplish increasingly difficult tasks using increasingly sophisticated robots. To confront this challenge, automatic categorization algorithms like the one proposed in this work will be indispensable.

References

Brooks R. A. (1986) A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation,* RA-2(1), pag. 14-26.

Brooks R. A. (1991) Intelligence without representation. *Artificial Intelligence* 47:139-159.

Brooks R. A. (1991b) Intelligence without reason. *MIT AI Memo 1293.*

Brooks R. A. (1992) Artificial Life and Real Robots. *Applied AI.*

Caruana R. (1993) Multitask Learning: A Knowledge-Based Source of Inductive Bias. *In Proceedings of the Tenth Int. Conf. on Machine Learning.*

Connell J. H. (1989) A behavior based arm controller. *IEEE Transaction on Robotics and Autonomous*. Vol 5, N. 6.

Dorigo, M., and Colombetti M. (1994) Robot Shaping: developing autonomous agents through learning. *Artificial Intelligence* 71:321-370.

Goldberg D. E. (1989) Genetic Algorithms in search, optimization and machine learning. *Addison Wesley*.

Kaelbling L. P., Littman M. L. and Moore A. W. (1995) An Introduction to Reinforcement Learning. In *"The Biology and Technology of Intelligent Autonomous Agents", Springer-Verlag.*

Koza J. R. (1992) Genetic programming: On the programming of Computers by Means of Natural Selection. *Cambridge, MA: MIT Press.*

Maes P. and Brooks R. A. (1990) Learning to Coordinate Behaviors. *Proceedings of the AAAI.*

Mahadevan, S. and Connell, J. (1992) Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence* 55:311-365.

Mataric M. J. (1990) Navigation with a rat brain: A neurobiologically-inspired model for robot spatial representation. *Proceedings of the First Int. Conf. on Simulation and Adaptive Behavior.* MIT Press.

Pfeifer R. (1995) Cognition - Perspectives from Autonomous Agents. In "*The Biology and Technology of Intelligent Autonomous Agents". Springer-Verlag*.

Sutton, R. S. and Barto, A. G. (1998) Reinforcement Learning: An Introduction. *A Bradford Book. The MIT Press.*

Utgoff, P. E. and Cohen, P. R. (1998) Applicability of Reinforcement Learning. In *The Methodology of Applying Machine Learning: Problem Definition, Task Decomposition and Technique Selection Workshop*, ICML-98. pag. 37-43.

Watkins C. J. C. H., Dayan P. (1992) Q-Learning. *Machine Learning,* 8:279-292.