D. Cliff, P. Husband, J.A. Meyer and S.W. Wilson (Eds.). Cambridge, MA: The MIT Press/Bradford Books. pp. 144-153.

A Hierarchical Classifier System Implementing a Motivationally Autonomous Animat

Jean-Yves Donnart and Jean-Arcady Meyer
Groupe de BioInformatique
Ecole Normale Supérieure, CNRS-URA 686
46, rue d'Ulm 75230 Paris Cedex 05, France
(donnart@wotan.ens.fr - meyer@wotan.ens.fr)

Abstract

This work describes a control architecture based on a hierarchical classifier system. This architecture, which uses both reactive and planning rules, implements a motivationally autonomous animat that chooses the actions it will perform according to the expected consequences of the alternatives. The adaptive faculties of this animat are illustrated through various examples.

1 Introduction

The behavior of an animat is adaptive to the extent that this behavior allows the animat to "survive," that is to maintain its *essential variables* within their *viability zone*, even when confronted with a changing environment ([Mey91a]). According to present knowledge on animal behavior ([McF93], [Mel94]), it may be supposed that several complementary components or mechanisms can be called upon in the design of an adaptive animat. Besides the fact that it would appear necessary to equip the animat with sensors and actuators and to interconnect these through an equivalent to an appropriate nervous system, further adaptive properties can be gained from the use of a memory, a motivational system and a planning system. Likewise, it may prove effective to hardwire some components or mechanisms, while leaving others free to change in the course of a learning process. The work described in this paper puts these ideas into practice and begins by describing the architecture of an animat with adaptive capabilities that depend, among other things, on a hierarchical classifier system. These capacities are illustrated in the specific context of a navigational task. The architecture's main functionalities are then compared with a variety of creations or propositions arising out of ethology, computer science, and robotics. It turns out that it implements a *motivationally autonomous system* as described by McFarland and Bösser ([McF93]); it accordingly has the potential to exhibit, at least to a certain degree, the adaptive properties of the most advanced animals.

2 The Architecture

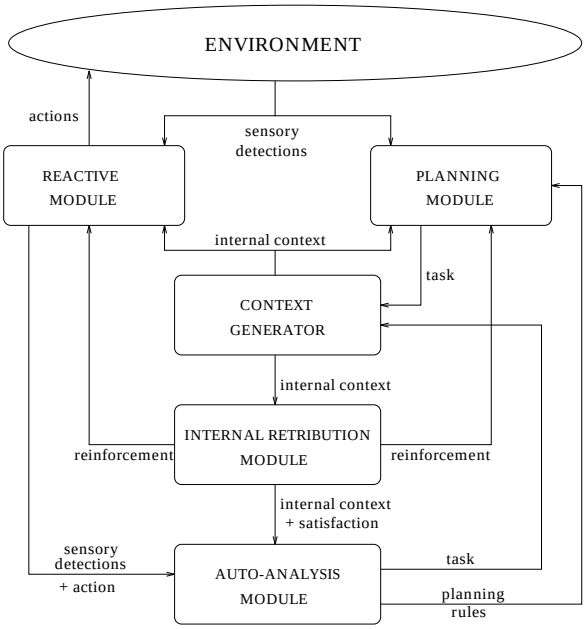

Figure 1: The architecture

In its current configuration, the architecture proposed here enables an animat to navigate from one point to another in a two-dimensional environment that may contain assorted materials, and obstacles in particular. A variety of possible generalizations of this implementation, capable of conferring on the animat many additional adaptive abilities, are in the process of realization and will be dealt with in this text. The animat is equipped with three proximate sensors that keep it informed of the presence or absence of material elements in front of it, 90° to its right, or 90° to its left. It is also able to estimate the spatial coordinates of the position it is located in and the direction of a goal to be reached in each of the eight sectors of the space surrounding it. Lastly, it is capable of moving straight ahead, 90° to its right,

or 90° to its left. This architecture, which relies upon a hierarchical classifier system ([Hol86]), is organized into five modules - a reactive module, a planning module, a context generator, an internal retribution module and an auto-analysis module (Figure 1).

2.1 The Reactive Module

This module consists of a series of rules - or classifiers - that allow the animat to react to incoming sensory information from the environment as well as to the internal context generated by the context generator. In the current configuration, the internal context is the direction of the current goal. The reactive rules take the form:

If <sensory information> and <direction of current goal> then <action>

For example, rule *R1 :* $100|001 ==> 01$ can be activated when the animat becomes aware of the presence of a material element in front of it, but not on either side (information coded by "1", "0" and "0"), and when the direction of the current goal is 45° to its right (direction "001"). If this rule is actived, the animat performs an elementary displacement 90° to its right (action "01").
An internal strength - which will be discussed later - is associated with each rule of the reactive module, as is a prediction of the consequences of the corresponding action. This prediction is learned incrementally and describes how the animat's position changes when the rule is applied. For instance, the prediction $X := X + 1$, $Y := Y$ will be associated with rule *R1*.
To each pair of conditions on the *<sensory information>* and the *<direction of current goal>* correspond, at any point in time, three rules capable of being triggered, each of which is associated with one out of the three possible actions. The choice of which rule is actually actuated is effected probabilistically, on the basis of the respective strengths of the three rules involved. In the current version of the system, there are $8 * 8 * 3 = 192$ possible rules. These rules are all created when the system is initialized and charged into the reactive module, which does not change in size as long as the system is in operation. The strengths of each rule are all initialized to a given value and are subsequently modified by learning. In the future, a genetic algorithm ([Hol86]) will be used in the interest of discovering more general rules.

2.2 The Planning Module

This module is made up of another series of rules, that allow it to decompose a task[1] into a sub-task according to current sensory information. These rules take the form:

If <sensory information> and <current task> then <new current task>

Sensory information is provided by the sensors and comes from the environment. It consists of information supplied by the proximate sensors and of the animat's coordinates and current orientation. As explained below, the *<current task>* is the one registered at the top of the pile of tasks governed by the context generator. The current task is used as an internal context which contributes to the triggering of the planning rules. The *<current task>* and the *<new current task>* are each coded at present as a pair of coordinates. Later, they will be described in a more general form.
Thus, the rule *P1 :* $000|5,1|001|3,0;3,5 ==> 5,1;5,2$ can be activated at point 5,1 (in (x,y) spatial coordinates) if the animat is headed in a north-easterly direction (coded by "001"), if it perceives no material element ahead or on either side of it (coded by "0", "0" and "0"), and if its current task is to reach the point with coordinates 3,5 when starting at point 3,0. If this rule is activated, the new task, which involves reaching point 5,2 from point 5,1 will be placed on top of the context generator pile.
With each rule of the planning module are associated two strengths - one local, the other global - the evaluation of which will be explained later. The local strength is used to determine the probability of triggering a rule whose condition part matches the current situation. When the system is initialized, the planning module is empty. During operation, this module can dynamically acquire rules - generated by the auto-analysis module - or lose rules - according to how the global strengths of these rules evolve. The size of the planning module thus varies over time, though it cannot exceed a preestablished upper limit.

2.3 The Context Generator

The context generator consists essentially of a pile of tasks the top of which represents the current task of the system. New tasks are added to the pile either by the planning module, as just seen, or by the auto-analysis module, when an obstacle is detected. This latter case occurs when the presence of an obstacle prevents the action judged to be the most conducive to reaching the goal associated with the current task from being fully applied. An obstacle can be detected in two ways:
1) When a rule is triggered by the reactive module and the results obtained are not those expected (a situation designated as failure).
2) When the system can predict that a rule will lead to failure if it is applied.

[1] In AI, this type of decomposition is classically called problem decomposition. Wilson ([Wil87]) discusses a decomposition of behavior modules. The term task appears more general and is derived from robotics terminology ([Alb81]).

For example, rule *R2 :* 100|110 ==> 00 - that stipulates that, in the presence of a material element situated in front of the animat (coded by "1", "0" and "0") and of a goal situated 45° to its left (direction coded by "110"), it should proceed straight ahead (action coded by "00") - may have been triggered because it presented the strongest strength at a given moment. This rule predicts that $X := X$ and $Y := Y + 1$. However, because the material element is actually an obstacle, Y cannot change. The obstacle is therefore recognized as such by the auto-analysis module, which generates a general subtask. This *"skirting around the obstacle"* task is to go beyond the line which lies perpendicular to the direction of the animat and passes through the point where the obstacle has been detected (Figure 2a). This task is coded by the pair <*coordinates of the point*> <*direction vector of the straight line to be crossed*> and is registered at the top of the pile of the context generator.

The context generator also includes an algorithm that transforms the current task, situated at the top of its pile, into a goal, then supplies the direction of this goal to the reactive module. In the case of a task posted by the planning module, the corresponding goal is simply described by the coordinates of the point to be reached. In the case of a task posted by the auto-analysis module, the goal is described by the coordinates of the projection of the animat's current location on the straight line to be crossed. This projection, and accordingly the corresponding direction information, varies whenever the animat moves.

An *emergent functionality* ([Ste91]) of the internal dynamics of the system, and more particularly of the context generator, is to enable the animat to skirt around the obstacles it encounters. In fact, if we reexamine the Figure 2a example, when the obstacle is detected at point 1, the task of having to go beyond the straight line $\delta 1$ is placed on top of the current task - which corresponded to the goal direction *d0* - and the current goal becomes the point marked with a "?" on the figure, in direction *d1*. Since rule *R2* cannot be applied, the animat chooses to apply another, causing it, for example, to move towards the left. After this displacement, the animat arrives at point 2. The current goal, with which direction *d2* is associated, becomes that in Figure 2b but the pile of tasks doesn't change.

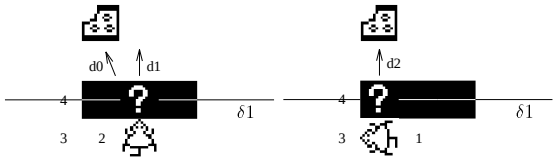

Figure 2: a / b

At point 2, if the learning has progressed sufficiently, the best action the animat can perform consists in progressing to the right in order to move in direction *d2*. As this action is not possible, it chooses to move forward. After arriving at point 3, the current goal becomes that of Figure 3a, in direction *d3*. This time, the best rule can be applied: the animat turns right and reaches the current goal. The task associated with $\delta 1$ is erased, and the animat can resume pursuing the goal of direction *d4* (Figure 3b).

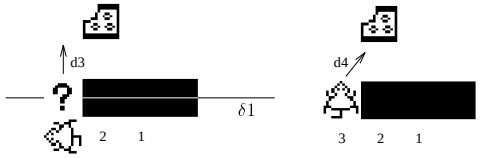

Figure 3: a / b

Following this reasoning, had the animat encountered an obstacle perpendicular to the preceding, preventing it for example from reaching point 3, this obstacle would have been detected as the preceding one had been. This is because the forward action, judged to be the best since turning right was already forbidden, has become impossible. In that case, the task of having to go beyond the line $\delta 2$ would have been placed on top of the task associated with $\delta 1$. The animat would thus have been directed successively through points 3 and 4 (Figure 4). At 4, it would have been able to turn right and reach point 5, where the task linked with $\delta 2$ would have been erased. Likewise, the task linked with $\delta 1$ would in turn have been erased at point 9.

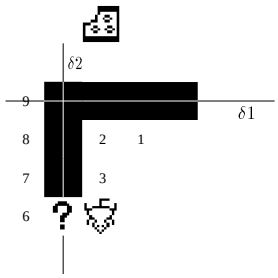

Figure 4:

In more complex situations, it can happen that the animat, seeking to attain the current goal, will erase a task placed farther down than the current task in the pile. In this case, all the tasks situated above the erased task are also erased, as they were generated for the sole purpose of executing this task and no longer have any justification. It will be shown later on that such mechanisms enable the animat to skirt around obstacles and to extricate itself from dead-ends with arbitrarily complicated shapes.

2.4 The Internal Retribution Module

The retribution module works through a process of reinforcement which causes the strengths of the rules of the reactive and planning modules to change.
Within the reactive module, this reinforcement takes place each time a rule is used. It depends on the *satisfaction* of the animat, that is, on an estimation of the success with which this rule brought the animat closer to, or took it farther from, the current goal:

$$S(R, u+1) = (1-\alpha) * S(R, u) + \alpha * satisf$$

$$satisf = \frac{dist\_goal(u) - dist\_goal(u-1) + max\_dist}{max\_dist * 2}$$

where
S (R, u) : strength of rule R after u triggers
S (R,0), α, max_dist : parameters of simulation
dist_goal(u) : estimated distance to the current goal.

In the present version of the system, distances are calculated exactly. However, distance information does not need to be very accurate because it is used merely as a heuristic for triggering good reactive rules, which are not necessarily optimal. It is the role of the planning module to orientate the reactive module towards a near optimal path, by means of the internal context.
A failure can occur when the action of a rule is executed. For instance, going forward when there is an obstacle in front of the animat is impossible. In that case, the strength of the rule will be reset to zero.

The rules of the planning module are characterized by two strengths: a local strength and a global strength. The local strength evaluates the usefulness of decomposing a task into a sub-task proposed by the rule. The global strength, on the other hand, detects and suppresses the rules which are unlikely to be used by the system.
To calculate the local strength of a rule *P1* that decomposes, for example, a task T - such as that of going from i to f - into a sub-task *T1* - such as that of going from j to k - the retribution module evaluates the average cost of all the paths tested by the animat which enable it to reach f from i without resorting to decomposition. In the present version of the system, this cost, denoted by *AC (T)*, is a average of the lengths of the paths expressed in terms of the number of elementary displacements they contain. *AC (T)* is evaluated incrementally:

$$AC(T, u+1) = (1-\alpha) * AC(T, u) + \alpha * C$$

where
u : number of times that task T was accomplished
C : cost of the u+1th path
α : parameter.

The retribution module also evaluates the average cost of all the paths joining i to f and using rule *P1*, that is, that reach k from j:

$$AC(P1, u+1) = (1-\alpha) * AC(P1, u) + \alpha * C1$$

C1 is computed when task T is erased, with the cost C of the path covered being divided among all the rules *P1, P2, ...* which decomposed T. This succession of rules is thus memorized, and a *profit sharing* algorithm ([Gre88]) is called on to manage the corresponding retributions. In these conditions, the local strength of *P1* is given by:

$$LS(P1, u+1) = \frac{AC(T, u)}{AC(P1, u)}$$

The shorter the paths joining i to f and passing through j and k have turned out to be on average than the paths joining i to f by way of other paths the stronger this strength is.
Each time a task generated by the planning module is at the top of the context generator pile, the planning module can decide whether or not to decompose this task and to trigger one of various decomposition rules *P1, P2, ...* it contains. These rules have been created by the auto-analysis module from salient states detected in the environment, as will be explained later. The decision to decompose is then made on the basis of a probabilistic choice depending on the local strength of each rule (a local strength of 1 being assigned to the non-decomposition option), weighted by an exploration-exploitation coefficient. The role of this coefficient is to modify, in the course of operation, the probability that the system will apply rules with a high local strength. Its value is constant for the time being; it could subsequently be modified according to the progress of the simulation.
It is clear that, whatever the local strength of rules *P1, P2,...* that decompose task T, these rules are unlikely to be triggered if task T itself is unlikely to be posted on the context generator pile. The global strength of each rule allows this type of situation to be detected and the less useful rules to be eliminated whenever the size of the planning module exceeds the maximum permissible threshold. The global strength of *P1* that decomposes T into *T1* is given by:

$$GS(P1) = LS(P1) * GS(T)$$

GS (T) is the average of the global strengths of all the rules P liable to post T on the context generator pile, this average being calculated incrementally:

$$GS(T, u+1) = (1-\alpha) * GS(T, u) + \alpha * GS(P, u)$$

When the system is initialized, a principal task PT - for example, going from point 3,0 to point 3,5 - is assigned to the animat, and the corresponding global strength *GS (PT)* is set arbitrarily at 1000. Under these conditions,

if task T happens to be generated only rarely, its global strength will be weak, as will be those of all the rules that decompose T.

2.5 The Auto-Analysis Module

Besides its role in the characterization of obstacles, the analysis module is used to detect recursively the *salient states*[2] of the environment. To accomplish this, the module calculates, from the satisfaction of the animat after each completed action, the variation of this satisfaction between two successives actions. In positions where the corresponding gradient is positive, the analysis module detects *satisfaction states*, which are added to the departure and arrival states of the path in question. These satisfaction states are only detected when the pile contains a "skirting around the obstacle" task.
The recursive process effected by the analysis module applies first of all to the path actually travelled by the animat, then to the successive fictitious paths that can be abstracted from the satisfaction states detected on these paths. Such a process is thus a consequence of the metaphor which conceives planning as a series of "thought experiments". When the path obtained by direct connection of the satisfaction states detected on the preceding path generates the same sequence of satisfaction states, the recursion is stopped, and the last satisfaction states discovered are recognized as salient states.

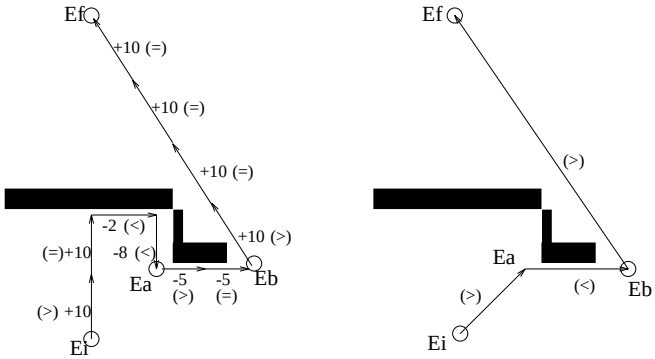

Figure 5: Satisfaction states: Numerical values indicate the satisfaction brought by each action. $>$, $<$ and $=$ symbols indicate the sign of the satisfaction gradient.

Thus, in the case of Figure 5, the animat has accomplished ten actions in order to reach state *Ef* from state *Ei*, while simultaneously skirting around an obstacle. The gradient of satisfaction is positive at *Ea* and *Eb* during a "skirting around" task, and the analysis module accordingly generates four satisfaction states: *Ei, Ef, Ea* and *Eb*. At the next stage, the satisfaction gradient associated with each fictitious action, which makes it possible to progress from one satisfaction state to the next, is computed. *Ea* can then be eliminated, as the gradient is negative between *Ea* and *Eb*. Because no other satisfaction point can be eliminated along the path directly connecting the three remaining states with one another, the recursive process stops at this stage, and the analysis module recognizes and memorizes three salient states: *Ei, Ef* and *Eb*, each characterized by its coordinates, by the sensory information obtained by the animat at this point, and by the corresponding orientation of the animat.
The salient states are then used to generate planning rules. Thus, at the conclusion of the path described on Figure 5, the two rules *P1* and *P2* that decompose path *Ei-Ef* into *Ei-Eb* and *Eb-Ef*:

P1: $000|3,0|000|3,0,3,6 ==> 3,0,6,3$
P2: $010|6,3|000|3,0,3,6 ==> 6,3,3,6$

are created and input to the planning module. The advantage of the redundancy in the description of these rules is to make it possible to recognize the salient states, even when there are imprecisions in the coordinates, or in the sensory information, or in the animat's orientation, or when the animat reaches one of these points with a new orientation.
A current weakness of the model is that absolute (x, y) coordinates are used to code the tasks. In the future, the position and the direction of each salient state will be calculated relatively to the position of the previous salient state of the plan. A "relative-position estimator", analogous to the one used by Kuipers [Kui93], will allow the animat to navigate from one salient state to another. Also, a more general description of salient states will be sought by means of a genetic algorithm.

3 Operation of the System

The following examples describe the results obtained in the instance of an animat moving in a square environment.
Figures 6, 7 and 8 illustrate the capacities of the animat, when learning only reactive rules, to skirt around obstacles of various shapes and to adapt to changing circumstances. The path on Figure 6 is the one obtained after 50 iterations - that is, after 50 experiments during which the animat has reached the goal when starting from the initial state - in an environment containing a dead-end. Path 7 is the one obtained when the dead-end is replaced by a double spiral, after five additional iterations in this new environment. Path 8 is the one obtained when the double spiral is replaced by an arc after a single additional iteration.

[2] In a purely navigational task, one could have used the word "landmarks" instead. We prefer to refer to "salient states" as our approach aims at solving more general tasks.

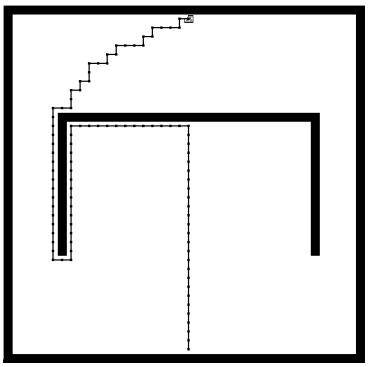

Figure 6: The dead-end environment

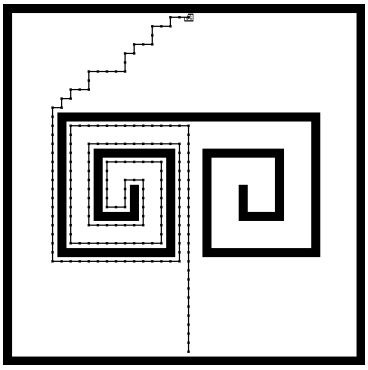

Figure 7: The double spiral environment

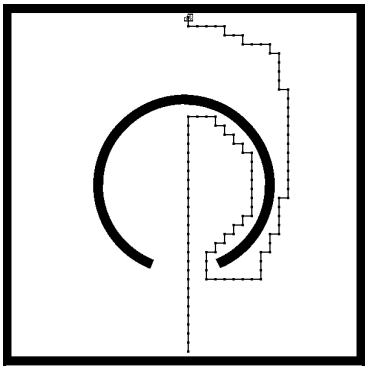

Figure 8: The arc environment

Figures 9 and 10 show what salient states and what corresponding action plan are found in the environments with a dead-end and with a double spiral, using the reactive paths from Figures 6 and 7. In both cases, it is apparent that the original task is decomposed into three sub-tasks and that the plan which is thus abstracted from the actual paths is simple because it depends on the convex envelope of the obstacles, rather than on the complexity of these obstacles.

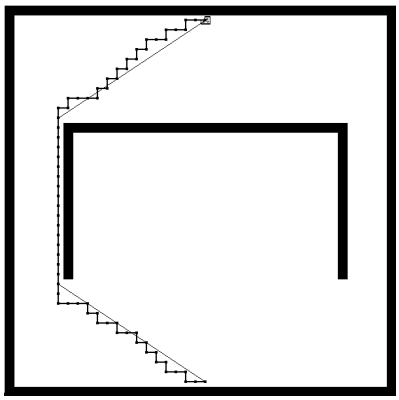

Figure 9: Plan abstracted in the dead-end environment, and a path obtained when using this plan

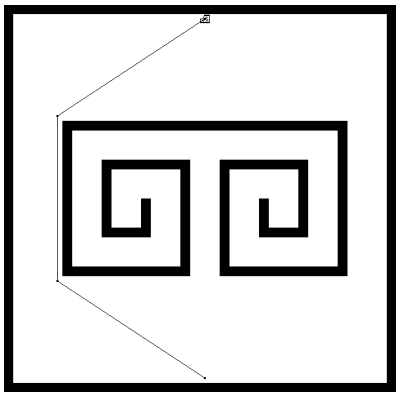

Figure 10: Plan abstracted in the double spiral environment

A path following the new plan is also shown in Figure 9. The three planning rules which define this plan are successively triggered, activating the three corresponding sub-tasks. At its starting point, the animat has used the first salient point as a subgoal, and when it arrived at this point, it used the second salient point as another subgoal. The planning module influenced the activation of the reactive rules in order to generate a shorter path to the goal.
Figures 11, 12 and 13 illustrate the decomposition of a plan into three hierarchical levels. The plan in Figure 11 is obtained from a path generated when using reactive rules only. When this plan is carried out, it is seen that the necessity of avoiding an obstacle yields a path that defines new salient states and that new sub-tasks are superimposed on the preceding (Figure 12). Likewise, when this new plan is put into effect, a third level of sub-tasks appears (Figure 13). Beyond this third level, the paths travelled no longer involve the avoidance of obstacles, and the corresponding plan does not result in additional decompositions.

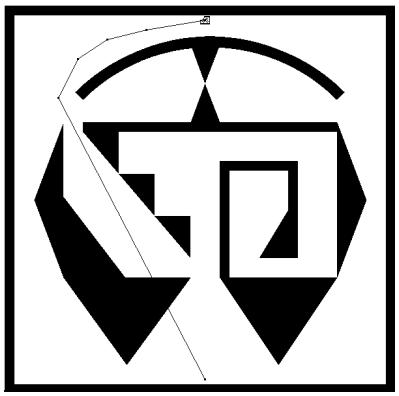

Figure 11: Plan level 1

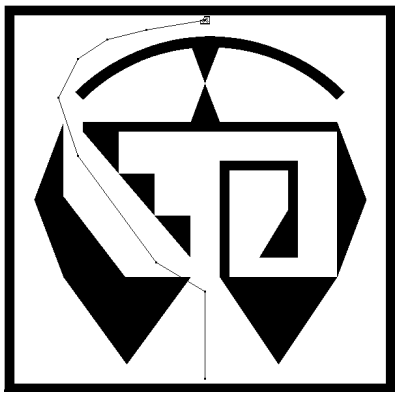

Figure 12: Plan level 2

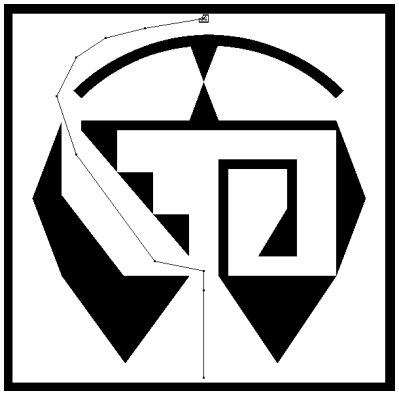

Figure 13: Plan level 3

Figures 14 to 16 illustrate the fact that the system retains several plans in its memory and that it is continually updating the local and global strengths of the rules of the planning module. It consequently follows that it can rapidly switch from one plan to another, or create new plans, as a function of the new obstacles appearing in the environment.

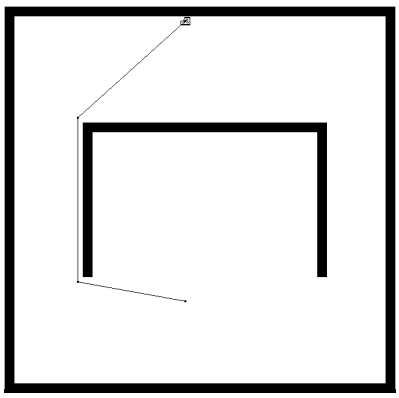

Figure 14: The best plan : iteration 15

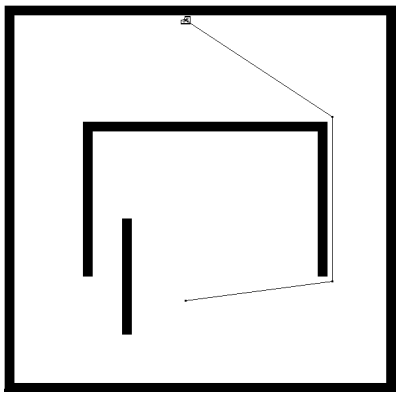

Figure 15: The best plan : iteration 22

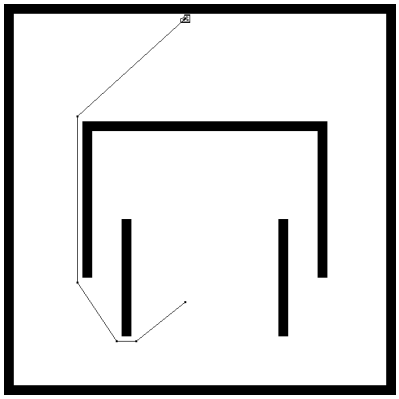

Figure 16: The best plan : iteration 38

Thus, after 15 iterations in the environment depicted on Figure 14, the animat has memorized two plans for

avoiding the dead-end. The best plan is shown on Figure 14, while the less effective one is shown on Figure 15. If, at iteration 16, a new obstacle is added to the environment along the animat's optimum path, this obstacle is avoided, and the corresponding plan is modified accordingly. However, as the cost of this modified plan exceeds the cost of the second plan stored in the memory, this second plan is the one most likely to govern the animat's path from the 22th iteration on. Likewise, introducing a new obstacle into the environment at the 30th iteration gives the advantage once again to the modified version of the first plan (Figure 16). It is thereby seen that the animat is capable of altering its plans as a reaction to modifications in its enviroment.

4 Related Research

As compared with related computer or robotics approaches, the architecture proposed here displays certain distinguishing characteristics.

Where learning is concerned, this architecture makes it possible to avoid the problem of the "two-platform station" described by Westerdale ([Wes89]) as applied to traditional classifier systems. In these systems, a single rule can be rewarded at times and punished at others, according to the corresponding context. Here, the context of use is expressly taken into account and is used to distinguish the rules. Likewise, the problem of the "temporal credit assignment" ([Sut91]), inherent to traditional classifier systems when a large number of rules have been involved in the acquisition of a reward, is eliminated here because the strength of each reactive rule is updated after its utilization, thanks to the management of an internal reward. This, however, is not at all the case with the local strength of the planning rules. Nevertheless, the corresponding profit sharing plan concerns only a given level of planning and therefore calls on only a small number of rules, in agreement with the logic envisaged by Wilson ([Wil87]) for the bucket brigade algorithm ([Hol86]).

The architectures described by [Wil87], by [Shu91] and by [Col93a] also rely on hierarchical classifier systems, but only the first - which is a theoretical construction and has not yielded any concrete application - might implement a planning process. Nevertheless, Wilson does not specifiy how the corresponding tasks and sub-tasks could be identified by the system. Likewise, though the architecture proposed by Colombetti and Dorigo provides for a classifier system to coordinate the actions proposed by two other classifier systems, the hierarchical relationships are predetermined by the programmer. On the contrary, in the present work, the hierarchical relationships among tasks are dynamic because they are generated internally on the basis of the experience gained by the animat.

As to planning, the architecture used here does not call on any predefined operators for decomposing problems into sub-problems, for the purpose of generating a plan which would then be executed ([Nil80]). Such a practice, which implies that planning precedes acting, has shown itself to be singularly ineffective ([Bro91]). Conversely, here, acting precedes planning, and the latter does not depend on predefined operators, but rather is abstracted from the paths actually travelled. The plans thus elaborated are initially quite general and are based on a small number of rules - thereby reducing the cost of learning, as was just mentioned. These plans are refined as needed. They are not executed mechanically by the animat, but instead are used as one ressource among others to decide which action to perform ([Agr88], [Suc87]). The organization of these plans thus appears as an emergent property, arising from the interactions between the animat and its environment and elicited by the animat's needs. Lastly, the value of these plans is continually reevaluated, which confers considerable adaptive faculties on the system.

Albus ([Alb81]), too, described a hierarchical architecture able to decompose a complex task into a series of sub-tasks, then into a series of elemental moves, then into a series of motor drive signals which actuate observable behavior in a robot. However, although Albus describes how such architecture relates to a general theory of intelligence ([Alb91]), he doesn't state how the corresponding hierarchy might be dynamically generated, nor how it could be modified according to the robot's needs and to the environmental conditions encountered.

In comparison with the literature on animal behavior, the architecture proposed here implements a *motivationally autonomous agent* ([McF93]). In everyday language, the term motivation is used to describe the experience of desiring to act in particular ways in order to achieve certain ends. As Toates ([Toa86]) argues, a motivational system is one that selects a goal to be pursued and organizes the animal's commerce with it. Current research on the relationships between motivations and behavior focus on the notions of motivational space and motivational state ([McF85]). The motivational state of an animal at any particular time depends on its physiological or internal state, on the cue state arising from its perception of the external world, on the consequences of its current behavior and on the expected consequences of its future behavior. Such a motivational state can be portrayed as a point in a motivational space, the axes of which are the animal's important motivational stimuli, such as an energy level or the odor of food. As a consequence of the animal's behavior, the motivational state changes, and the corresponding point describes a trajectory in the motivational space. Such displacements are monitored by sensors that relay nervous messages to the brain. They are also calibrated in terms of their estimated contribution to the animal's fitness ([Hou76], [McF81]). Then a decision is made about which behavior

to perform. If this decision doesn't take into account the expected consequences of the alternatives, the animal behaves like a *motivated automaton*, otherwise it behaves like a *motivationally autonomous agent* ([McF93]). To do so, the animal requires knowledge of the probable consequences - or expected utility - of its acts. In other words, it must have some memory of the past consequences of similar activities, and it must be capable of planning - i.e. it must use some form of cognition. Furthermore, as Dennett ([Den83]) pointed out, it must want something, it must have goals[3].

The animat described here displays all these characteristics. Indeed, the behavior, or the action, it performs at any instant depends both on sensors and on what was called here the internal context. This context actually includes the goals that the animat has selected and that it seeks to achieve. There is nothing to prevent this context from subsequently taking into account other information about the internal state of the animat. The animat's goals are generated by an explicit planning process, and the strength of the rules memorizes the consequences of the various choices that the animat has made in the past. These consequences are actually evaluated in terms of their aptitude in bringing the animat nearer its goal; they may later depend on an appropriate utility function. It will be noted in passing that McFarland and Bösser do not specifiy how the weighting coefficients associated with the different actions can induce the animal to choose one action when pursuing one goal and another action when pursuing another goal. Here, this problem is solved by the fact that the actions performed depend on the internal context and accordingly on the current goal.

This latter point is likewise to be compared with the hypothesis proposed by Wilson ([Wil91]) according to which the most efficient systems would be those that "convert every frequently encountered important situation to one of 'virtual stimulus-response' in which internal state (intention, memory) and sensory stimulus together form a compound stimulus that immediately implies the correct next intention or external action". As Wilson remarks, such an hypothesis is compatible with the observation that, "in animals and people, even complex behavior, if frequent and important enough, tends to become reflexive". It should also be related with the distinction between "interpreted" and "compiled" knowledge of the standard AI ([Lai86]).

In comparison with other approaches aimed at including a motivational system in the architecture of an animat ([Bee90], [Cec93], [Gab93], [Hal91]), this approach is the only one that incorporates a planning process which, as seen previously, substantially enhances the adaptive faculties of the animat. It would accordingly seem that, in the continuum described by McFarland and Bösser ([McF93]) which distinguishes motivated automata from motivationally autonomous agents, these other approaches tend to be situated in the former category, while the present approach would belong to the latter.

It is moreover clear that the animat's *behavioral sequences* described here are not random, which could be demonstrated using the same methods that ethologists do ([Gui86]). These sequences are organized according to the animat's goals, so that a given action tends preferably to be followed by one action in the context of a particular goal and by another action in the context of a different goal. Such an organization is by no means arbitrary but rather tends to maximise the utility function, contrary, for instance, to what is learned in Colombetti and Dorigo ([Col93b]). Nor is it determined once and for all, and it does not preclude reacting opportunistically to the surprises of the environment.

5 Conclusion and Ongoing Research

The architecture described in this text is that of a motivationally autonomous agent. It accordingly could prove useful in reproducing at least certain adaptive abilities of the most advanced animals, that have amply proven their aptitude to survive in more or less predictible and more or less threatening environments.

This work continues in three directions. The first aims at adapting this architecture to an actual robot and to the corresponding constraints. The second aims at improving the present implementation by managing other motivations and other actions - such as eating, drinking, or exploring the environment. Another improvement will allow the animat to generalize the knowledge it acquires - in order notably to permit the animat to avail itself, in new contexts, of isolated actions or behavioral sequences which proved useful in the context in which they were learned. Finally, the third direction seeks to demonstrate the generality of the approach described here by applying the corresponding architecture to various planning problems like those of the *block-world* ([Nil80]).

References

[Agr88] Agre, P.E. and Chapman, D. (1988) "What are Plans for ?" *MIT Dept. of Computer Science, Tech. Rep. 1050*

[Alb81] Albus, J.S. (1981) "Brains, behavior and robotics." *Byte Books.*

[Alb91] Albus, J.S. (1991) "Outline of a Theory of Intelligence." *IEEE Trans. Syst. Man and Cybernetics. 21, 3, 473-509.*

[3] In other words, the animal must be goal-achieving and goal-seeking. Whether his behavior is goal-directed or intentional is another issue ([Den83], [McF89]).

[Bee90] Beer, R.D. (1990) "Intelligence as Adaptive Behavior: an Experiment in Computational Neuroethology." *Academic Press.*

[Bro91] Brooks, R.A. (1991) "Intelligence without representations." *Artificial Intelligence 47, 139-159.*

[Cec93] Cecconi, F. and Parisi, D. (1993) "Neural networks with motivational units." *In [Mey93].*

[Col93a] Colombetti, M. and Dorigo, M. (1993) "Learning to Control an Autonomous Robot by Distributed Genetic Algorithms." *In [Mey93].*

[Col93b] Colombetti, M. and Dorigo, M. (1993) "Training Agents to Perform Sequential Behavior." *Adaptive Behavior. Under Press.*

[Den83] Dennett, D. (1983) "Intentional Systems in Cognitive Ethology: the 'Panglossian paradigm' defended." *Behavioral and Brain Science. 6, 343-390.*

[Gab93] Gabora, L.M. (1993) "Should I Stay or Should I Go: Coordinating Biological Needs with Continuously-updated Assesments of the Environment." *In [Mey93].*

[Gre88] Grefenstette, J.J. (1988) "Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms" *Machine Learning 3, p225-245.*

[Gui86] Guillot, A. (1986) "Revue générale des méthodes d'étude des séquences comportementales." *Etudes et analyses comportementales. 2(3): 86-106.*

[Hal91] Halperin, J.R.P. (1991) "Machine Motivation" *In [Mey91b].*

[Hol86] Holland, J.H. (1986) "Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems" *Machine Learning: An Artificial Intelligence Approach II.*

[Hou76] Houston, A. and McFarland, D. (1976) "On the measurement of motivational variables." *Animal Behavior. 24, 459-475.*

[Kui93] Kuipers, B. and al. (1993) "The Semantic Hierarchy in Robot Learning" *Robot Learning, Kluwer Academic Publishers.*

[Lai86] Laird, J.E. and al. (1986) "Chunking in Soar." *Machine Learning. 1, 11-46.*

[McF85] McFarland, D. (1985) "Animal behaviour. Psychobiology, ethology and evolution." *Longman Scientific and Technical.*

[McF89] McFarland, D. (1989) "The teleological imperative." *In Montefiore and Noble (Eds). Goals, No Goals and Own Goals. Unwin-Hyman.*

[McF93] McFarland. D. and Bösser, T. (1993) "Intelligent Behavior in Animals and Robots." *The MIT Press/Bradford Books.*

[McF81] McFarland, D. and Houston, A. (1981) "Quantitative Ethology: the State-Space Approach." *Pitman.*

[Mel94] Mel, B.W. (1994) "Animal Behavior in Four Components." *In Roitblat and Meyer (Eds). Comparative Approches to Cognitive Science. The MIT Press/Bradford Books. Under Press.*

[Mey91a] Meyer, J.A. and Guillot, A. (1991) "Simulation of adaptive behavior in animats: review and prospect." *in [Mey91b].*

[Mey91b] Meyer, J.A. and Wilson, S. (Eds) (1991) "From Animals to Animats" *Proceedings of the 1st Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books.*

[Mey93] Meyer, J.A., Roitblat, H.L. and Wilson, S. (Eds) (1993) "From Animals to Animats 2." *Proceedings of the 2nd Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books.*

[Nil80] Nilsson, N.J. (1980) "Principles of Artificial Intelligence." *Palo Alto , Calif: Tioga Pub. Co.*

[Shu91] Shu, L. and Schaeffer, J. (1991) "HCS: Adding Hierarchies to Classifier Systems." *Proceedings of the 4th Int. Conf. on Genetic Algorithms.*

[Ste91] Steels, L. (1991) "Towards a Theory of Emergent Functionality." *in [Mey91b].*

[Suc87] Suchman, L.A. (1987) "Plans and Situated Actions: The Problem of Human-Machine Communication" *Cambridge University Press.*

[Sut91] Sutton, R.S. (1991) "Reinforcement Learning Architectures for Animats" *in [Mey91b].*

[Toa86] Toates, F.M. (1986) "Motivational systems." *Cambridge Univ. Press.*

[Wes89] Westerdale, T.H. (1989) "A Defense of the Bucket Brigade" *Proc. of the 3rd Int. Conf. on Genetic Algorithms.*

[Wil87] Wilson, S.W. (1987) "Hierarchical Credit Allocation in Classifier System" *in IJCAI 87.*

[Wil91] Wilson, S.W. (1991) "The animat path to AI" *in [Mey91b].*