NORTH-HOLLAND

Learning to Compose Fuzzy Behaviors for Autonomous Agents*

Andrea Bonarini and Filippo Basso
Politecnico di Milano AI and Robotics Project,
Dipartimento di Elettronica e Informazione,
Politecnico di Milano, Milano, Italy

ABSTRACT

We present S-ELF, an evolutionary algorithm that we have developed to learn the context of activation of fuzzy logic controllers implementing fuzzy behaviors for an autonomous agent. S-ELF learns context metarules that are used to coordinate basic behaviors in order to perform complex tasks in a partially and imprecisely known environment. Context metarules are expressed in terms of positive and negated fuzzy predicates. We also show how S-ELF can learn robust and portable behaviors, thus reducing the time and effort to design behavior-based agents. © *1997 Elsevier Science Inc.*

KEYWORDS: ***reinforcement learning, fuzzy control, autonomous agents***

1. INTRODUCTION

Since Brooks's first seminal papers [11, 12], many autonomous agents have been implemented following the *behavior-based* paradigm, where the *behavior* of an agent comes from the composition of *basic behaviors*, in principle independent of each other. This is considered a successful design practice, according to the principle of *problem decomposition.*

Address correspondence to Andrea Bonarini, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy. E-mail: bonarini@elet.polimi.it.

*We would like to thank A. Saffiotti, who suggested important issues during interesting discussions. This work has been partially supported by the MURST Project 60% "Development of autonomous agents through machine learning."

Received September 1, 1996; accepted December 1, 1996.

International Journal of Approximate Reasoning 1997; 17:409–432

655 Avenue of the Americas, New York, NY 10010
0888-613X/97/$17.00
PII S0888-613X(97)00002-9

Among the possible implementations for behaviors, the most common are finite-state machines, following Brooks's original approach [11, 19, 18]; classifier systems [13, 23]; and fuzzy rules [21, 2, 4, 6].

The combination of the different basic behaviors is often obtained through the application of *inhibition* mechanisms, in some cases integrated in the *subsumption architecture* [11, 19, 18]. The adoption of a fuzzy logic representation makes possible also other forms of interaction, such as *effect combination* [21, 3].

One of the main problems for the behavior-based approach to agent design concerns the identification of the combination of the most suitable basic behaviors to achieve a task in a given situation. Some approaches to support this design phase by machine learning have been proposed. Among the others, Mahadevan and Connell [18] propose a system that learns the basic behaviors most suitable for a given, predefined behavior architecture. The system of Dorigo and Colombetti [13] learns behaviors organized in different hierarchical architectures. That of Bonarini [3, 6, 8] learns a coordinator, implemented by fuzzy rules, that weights the output of basic behaviors.

In this paper, we present S-ELF (symbolic ELF), a *reinforcement learning* [17] system that learns to coordinate predefined basic behaviors by identifying the best *contexts* for each of them. We have developed S-ELF from ELF (evolutionary learning of fuzzy rules) [2, 4], a system that we have successfully adopted in the past to learn *fuzzy behaviors* (i.e., behaviors implemented by *fuzzy rules*), and their coordination [3, 6, 8].

S-ELF learns the context of activation for each available basic behavior. It works on contexts described by logical expressions composed by conjunctions of both positive and negative high-level fuzzy predicates, such as "*(in corridor-1)*" or "*(face door-2)*." S-ELF produces fuzzy metarules that relate each behavior to the best contexts in which it can be applied. A standard fuzzy composition mechanism then combines the different behaviors in contexts described by different sets of fuzzy predicates.

We have tested S-ELF on a control architecture similar to that of Flakey [21], where navigation is programmed in terms of fuzzy behaviors and their contexts of activation. Learning a control system described by high-level predicates brings about robust behaviors that can be instantiated in different environments, as we show with our experiments.

In this paper, we first present the conceptual framework we have adopted to represent fuzzy behaviors, and the architecture of the fuzzy control system that we have implemented (Section 2). Then we describe the main features of S-ELF, focusing on some learning mechanisms that may be interesting for any reinforcement learning algorithm (Section 3). Finally, we discuss the experimental results that we have obtained by applying S-ELF in the framework described in Section 2.

2. FUZZY BEHAVIORS AND COORDINATION

2.1. Fuzzy Behaviors

Following the framework proposed in [21], we have defined a *behavior* as a triplet

$$\langle C, BB_A, O \rangle$$

where:

- C is the *context of application* of the behavior, that is, a description of the situations where the behavior should be applied;
- BB_A is a function that computes, for each state-action pair $\langle s, a \rangle$, how *desirable* it is to perform the action a when the agent is in the state s, in order to realize the basic ability A (for instance, from BB_A we have a measure of how desirable it is to *turn left by 20 degrees* when there is an obstacle *in front* of the agent in order to achieve the basic ability *follow the corridor*);
- O is the *object* with reference to which the basic ability A is realized (in the mentioned example, *corridor-1*).

We can say that BB_A represents *how* to implement a basic ability, C *when* to do it, and O the *element* of the environment on which the ability is applied.

As mentioned before, the information from the *desirability function* is more flexible, complete, and useful than that from a mapping between a state and the best action the agent can do in such a state. In fact, besides stating which is the best action in each state, it gives a grade, to any possible action, so enabling a composition of *desirability functions* related to different behaviors active at the same time. For instance, the desirability function can be implemented by means of a set of rules, of which the ith is

$$IF\ (state\ IS\ S_i)\ THEN\ (action\ IS\ A_i),$$

where both the antecedent and the consequent are fuzzy sets. In such a case, to each rule can be associated a function $D_{S_i \to A_i}(s, a)$, given by

$$D_{S_i \to A_i}(s, a) = \mu_{S_i}(s) \otimes \mu_{A_i}(a),$$

where $\otimes$ is a *T-norm*.

The *desirability function* BB_A of the behavior is given by

$$BB_A = \bigoplus_{i=1,\ldots,n} D_{S_i \to A_i}(s, a)$$

where n is the number of rules that implement the basic behavior, and $\oplus$ is a *T-conorm*.

The *context of application C* is defined by means of a logical combination of predicates. In Saffiotti's work [21], the *context of application* of a behavior is hand-coded: therefore, the designer is responsible for its correct implementation. It is of interest to understand whether it is possible to design an algorithm that automatically learns such contexts. Reinforcement learning algorithms suggest themselves for this application, since the only knowledge they need from the environment is an evaluation of the agent's performance. S-ELF is a reinforcement learning algorithm for the contexts of behaviors coded according to Saffiotti's framework.

Before explaining the algorithm in more detail, we can note that the situations in which a behavior has to be applied depend both on the agent's overall task and on the environmental conditions the agent has currently to face. In other words, it is possible to distinguish two conceptually distinct classes of predicates involved in the context definition:

- Some predicates are intimately coupled with the task the agent has to accomplish: for instance, the utility of the behavior *follow-corridor* is different if the task is to reach a point at its end or to enter the next door on the right. It is easy to see that such predicates are linked to the above definition of *mission*.
- Some other predicates represent the interaction of the agent with the environment: for example, the action computed by the behavior *avoid-obstacles* depends on current obstacle positions as detected by the sensorial apparatus.

We call the part of context defined by the first kind of predicates *global context* (GC), and we call the part linked to the environmental predicates the *environmental context* (EC).

Behavior activation is implemented by rules such as

$$IF\ (state\ s\ IS\ in\ context\ C)\ THEN\ (apply\ behavior\ A)$$

where $C = EC \cap GC$.

In order to make more evident the distinction between global and environmental contexts, we rewrite such rules as

$$IF\ (state\ s\ IS\ in\ environmental\ context\ EC)$$
$$THEN\ (apply\ behavior\ A)$$
$$WHEN\ (state\ s\ is\ in\ global\ context\ GC)$$

As mentioned before, the global context is closely related to the current high-level goal of the agent, i.e., to its mission. In our present implementation, missions are mutually exclusive; in other words, their *applicability conditions* (implemented as logical combinations of crisp predicates) define disjoint sets of situations. It is easy to understand that the applicability

conditions are the mission-level equivalent of the global context at the rule level. In other terms, the global context involves only crisp predicates, called *mission switches*: they are also responsible for the selection of the current mission. We can think of the rule base as composed of conceptually distinct subsets: to each subset belong all and only the rules sharing the same global context; they are only responsible for the accomplishment of a particular task, i.e., the development of a particular mission.

It is possible to replace the part

WHEN (*state s IS in global context GC*)

with

WHEN (*mission is M*)

where M is the mission that occurs when the state s is in the global context GC.

An example of a behavior activation rule is

IF (*at door-1*) *AND* (*closed door-1*)

THEN (*pass door-1*)

WHEN (*in room-1*)

where the antecedent of the rule corresponds to the environmental context, BB_A, is referred to by the name *pass*; O is *door-1*; and (*in room-1*) is the mission the rule refers to.

It is now possible to express better the aim of S-ELF. It learns which is the best environmental context for each basic behavior, given a mission to accomplish. In other words, S-ELF is able to learn the environmental context of a behavior, while its global context must still be hand-coded. This is not an important limitation: the definition of the global context is an easy task, because it involves only a very small number of crisp predicates, while the environmental context is defined by a larger number of fuzzy predicates.

To summarize, S-ELF is not able to select the current agent's mission, but, given a mission, it learns how to coordinate the basic behaviors to obtain it. To do so, it evolves a number of sets of rules, each one referring to a particular mission, such as

IF (*at door-1*) *AND* (*closed door-1*) *THEN* (*pass door-1*)

The environmental context of the behavior mentioned in the *THEN* part is obtained as the disjunction of the *IF* parts of all the rules sharing the same consequent.

Such rules work at a hierarchically higher level than the ones implementing a basic behavior, so they are *metarules* for the activation of a behavior. To avoid confusion, in the remainder of this paper, we will call them *metarules* or *context rules*, to distinguish them from the *control rules* implementing a basic behavior.

In the next section, we will describe in more detail how the control rules are implemented and how the context can *shape* them.

2.2. Context and Basic Behaviors

Even if the above-mentioned fuzzy metarules (or *context rules*) are, from the syntactical viewpoint, fuzzy implication relations [14], they are not implemented by using one of the standard fuzzy implication operators (e.g., T-conorms or pseudo inverses of T-norms, as in [21]). Instead, the truth value of the context in the current situation (i.e., the truth value of the *IF* part of the metarule) shapes the membership functions of either the antecedents or the consequents of the control rules that implement BB_A. In other terms, the fuzzy sets adopted in the control rules are dynamically modified by the context in a way described by the context rules. Thus, the context influences the interpretation of data or control actions in the control rules.

For a better understanding of this topic, further details on the shape of our control rules are needed. In our implementation, each basic behavior is implemented by means of only one control rule:

$$IF\ (s\ IS\ S)\ THEN\ (a\ IS\ A)$$

where both the antecedent and the consequent are fuzzy sets, and the fuzzy implication is implemented as a T-norm [14].

In particular, the fuzzy set S is defined over a set of values of a convenient state variable, and it can be seen as a *fuzzy predicate* representing some properties of the environment where the agent operates. We consider two such properties: the usefulness of having a given relationship with some input variable (*desirability*), and the possibility of doing so, according to environmental obstacles (*viability*). To be more concrete, we can imagine considering the *heading* of the agent as an input variable (say α). The properties we associate to α are the usefulness of going along α in order to realize the basic behavior itself (desirability), and the possibility of doing so, given the environmental situation (viability). Both the desirability and the viability are integrated in the antecedent predicate S, which is implemented as the intersection of two fuzzy subsets ($S = D \cap V$): D represents the desirability, and V, the viability. The desirability is a measure of how much the agent *would like* to be in a specific state s

(described by α) in order to reach its current goal, by adopting the action A described in the consequent of the control rule, while the viability is a measure of the extent to which it can really perform that action given the current environmental situation (e.g., obstacles and/or occlusions along the desired path).

For instance, in Figure 1 we show on the left the desirability and viability of being oriented in given direction in order to follow a corridor, when the agent is in the situation shown on the right. It is most desirable that the agent be oriented in the direction of the corridor, whereas, given the situation shown on the left of the figure, it is more viable to have a heading smaller than -10 degrees, that is, to turn left with respect to the present position.

As can be easily seen, D is related to a global viewpoint concerning the agent's mission, while V is linked to a local viewpoint about the detected world.

In a similar way, we define the fuzzy set A over a (crisp) set of actions that the agent can perform in the world; A yields all the information available about the agent's actions. Such information depends on the specific control variable that is selected. In our example, the effector variable is the *steering angle* (or *direction of movement*) of the agent: the information about this action is linked to its *utility* and *possibility*. The fuzzy set A is defined as the intersection of two fuzzy subsets: $A = U \cap L$. The first one (U) represents the utility of the specific action for reaching the goal, while the second (L) is related to the *limitations* that the mechanical and physical constraints of the actuators impose on the agent's actions (e.g., the maximum steering angle).

In the appendix, we will describe how the context of application of a behavior acts on the control rules that implement it. We have adopted a kind of interaction different from the ones proposed in the literature. The only approach, to our knowledge, in which the context of application of a behavior is defined via logical predicates and operations is Saffiotti's; in that work, the context acts on control rules by means of a pseudoinverse of

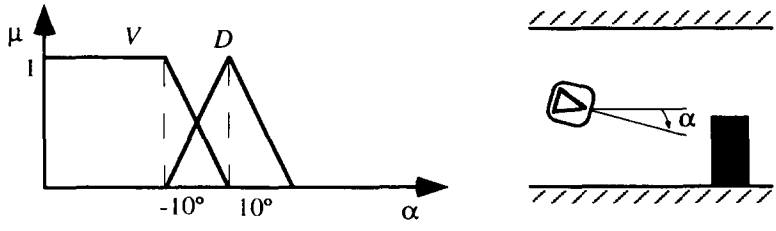

Figure 1. S, D, and V for the variable α.

a T-norm. In our implementation, the truth value of the context shapes the membership functions either of the antecedents or of the consequents of the control rules. In the appendix, we report details about how this is implemented in S-ELF.

3. MISSION

To take account of the different tasks an agent must face during its activity, we have introduced the notion of *mission*. It is a data structure that links a particular goal of the agent to the strategy of coordination of simple behaviors that can accomplish it. More formally, a mission is defined as an ordered 4-tuple

$$\langle G, AC, O, CS \rangle$$

where:

- G is the *goal* of the mission, that is, the task the mission allows the agent to accomplish;
- AC are the *applicability conditions* and code all the situations in which the mission has to take place;
- O is the *object* with reference to which the mission is developed;
- CS is the *strategy of coordination* of simple behaviors that realizes the mission.

For example, imagine that the mission is to follow the corridor labeled *corr-1*; this can be represented as follows:

- the *goal* G is to follow a generic corridor;
- the only *applicability condition* AC for the agent is to be in the corridor;
- the object O gives the reference to the specific instance of the generic corridor mentioned in G, here the one labeled *corr-1*.

In other words, G, AC, and CS depend on O, which plays the role of a variable; if O is not instantiated, then the defined structure is considered a *mission template*: a particular mission can be instantiated by assigning a value to O. In this way, the same abstract strategy of behavior coordination allows for the realization of the same type of mission in different environments.

Two observations can be made now. First of all, the applicability conditions of a mission depend on the overall task of the agent: for example, the mission *follow a corridor* may be useful for reaching a point at its end, but not for turning into the first door on the left. Second, the definition of a mission rests on the existence of a set of behaviors that we consider as *basic*. We must take care to choose them so that we are confident that their cooperation will allow the agent to accomplish the

kind of tasks we want it to face. Under this hypothesis, to learn a strategy for behavior coordination plays an important role, as we have demonstrated with S-ELF.

3.1. A Fuzzy Control Architecture Based on Coordinated Behaviors

Figure 2 shows the architecture of the system we have implemented. We can recognize four main modules:

- The *learning system* implements S-ELF, the learning algorithm that we describe in the next section; it continuously updates a rule base containing the fuzzy context metarules.
- The *coordinator* computes the activation level of all the behaviors that are to be fired in the current episode. To do so, first of all, it selects the current mission, according to the truth value of some binary predicates, called *mission switches*, supplied by the *switch generator SG*, which computes them from the sensorial observations. Then, the behavior activation levels are computed with reference to a subset of the rule base, composed of the rules selected for firing among the *matching rules* (i.e., the rules that match the current world state as it

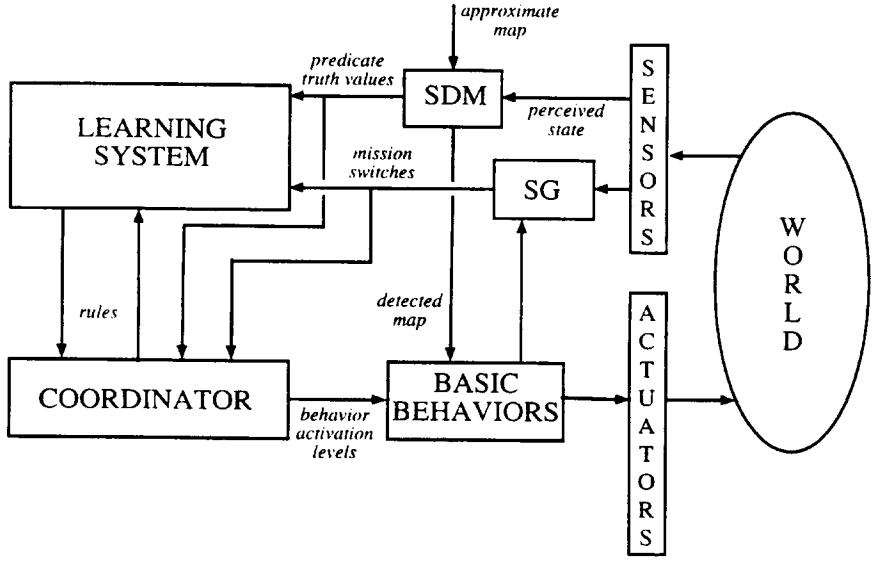

Figure 2. The system architecture.

has been perceived). In particular, during learning, not all the matching rules are fired, to make more evident the contributions of a small number of rules to the emergent behavior of the agent.

- The *basic behaviors* can be seen as a collection of operating modules that can act in parallel. Their activation and coordination is performed by the *coordinator*, which computes their activation level according to the sensed state and the current mission.
- The *sensorial data manager* (SDM) aggregates the information coming from sensors, and matches it with an *approximate map* of the environment where the agent has to act. In particular, the aggregation takes place at different levels of abstraction and interpretation, so that all the modules needing information about the surrounding world can find them at the required level of abstraction. The highest level of abstraction is implemented by a *detected map* that aggregates all the high-level information about elements of the environment, such as corridors and doors. Moreover, the SDM computes the truth values of the predicates used by the rule-matching algorithm.

4. LEARNING THE COORDINATOR

S-ELF is a reinforcement learning algorithm that operators on a population of fuzzy rules. Each member of the population is encoded by a chain of genes that represent the antecedent and the consequent of the rule. The antecedent of each rule encodes a *context* for a behavior, represented by the consequent.

The *antecedent* consists of n genes, one for each of the input predicates (*positional encoding*). Each gene may take a value in the set $\{1, 0, \#\}$, where 1 means that the corresponding predicate is considered as positive, 0 means that it is considered as negative, and # means that the predicate value is irrelevant for the characterization of the context for the behavior. Notice that, with this encoding, S-ELF can learn a complex description of a context, composed by conjunction of possibly negated predicates. For instance, the antecedent of the already mentioned rule

IF (*at door-1*) *AND* (*closed door-1*) *THEN* (*pass door-1*)

is encoded as ##1#0, where the third and fifth positions correspond to (*at door-1*) and (*closed door-1*).

The *consequent* contains only one gene, which denotes a basic behavior applied to an object [in the mentioned example, (*pass door-1*)].

4.1. Partitioning the Population

As in ELF [4], the rule population is partitioned into *subpopulations* in order to consider at each evaluating step only the controllers that have contributed to the agent's activity since the last evaluation. Thus, the competition among members of the population is local (it is limited to a *niche* [10, 23]). This improves the speed of convergence to a good solution by up to two orders of magnitude, compared with other proposals adopting genetic algorithms to learn fuzzy rules [4]. In ELF, each subpopulation corresponds to an antecedent configuration, since ELF should learn the best action for a given state. In S-ELF, we would like to learn the best context for a given basic behavior; thus, each subpopulation corresponds to a basic behavior, and the members of each subpopulation have different, competing contexts for the same behavior. Moreover, by adopting this partitioning, we have a relatively small number of subpopulations, since the number of basic behaviors is much smaller than that of the possible contexts.

4.2. Episodes

The evaluation of the performance of the agent is done at the end of a sequence of control steps, called *episode*. This produces some interesting effects [5]. If the episode ends when the agent reaches a particular (fuzzy) state, then the performance evaluation is done when something relevant happens, and, probably, it brings interesting information. In any case, this evaluation strategy averages the effects of the single rules, and, in general, it has a stabilizing effect.

At the beginning of an episode a subpopulation (i.e., a basic behavior) is selected, and during all the episode only rules belong to this subpopulation can trigger. At the end of each episode, the *reinforcement program* evaluates the agent's performance and distributes the corresponding reinforcement to the rules that have contributed to controlling the agent during the episode.

4.3. Reinforcement Distribution

To each rule is associated a measure of its estimated *strength*, i.e., of the estimated suitability of its antecedent to represent the context for the application of the basic behavior encoded by the consequent. The strength is updated at the end of each episode by a function that has the shape (common for reinforcement learning):

$$s(t + 1) = (1 - \alpha)s(t) + \alpha R(t),$$

where

- s is the strength of the rule;
- R is the reinforcement computed by the reinforcement program;
- α is the *learning rate*, computed in our case as

$$\alpha = \frac{\sum_{i=1}^{n_c} \textit{cycle-act-level}_i}{\min(\textit{enough-tested}, \sum_{i=1}^{n_a} \textit{cycle-act-level}_i)}.$$

Here:

- $\textit{cycle-act-level}_i$ is the activation level of the rule at control cycle i, i.e., how well its antecedent matches the state perceived at that cycle;
- n_c is the number of control cycles in the current episode;
- n_a is the number of control cycles to which the rule has given some contribution since its introduction in the rule base;
- *enough-tested* is a parameter that states when a rule has given enough contribution to the performed actions to be considered as tested enough, i.e., the rule *strength* can be considered a good estimation of the actual suitability of its antecedent to represent the appropriate context for the consequent.

In other terms, the value of α considers the fact that the rules can partially match a state, as is typical for fuzzy rules. So they receive a reinforcement that is proportional to the contribution they have given to reach the evaluated state.

At the end of each episode, reinforcement is also given to rules that have triggered in past episodes. Their strength is updated by

$$s(t+1) = (1-\alpha)s(t) + \alpha\gamma^k R(t),$$

where γ is a discount factor and k is the number of episodes between the current one and the one where the rule was triggered.

The relationship between this reinforcement distribution algorithm and *Q-learning* [22] is discussed elsewhere [5].

4.4. Rule Generation and Deletion

New rules are generated by the *cover detector* operator when the agent is in a state that is not matched by any rule. In this case, a new rule is generated for the selected subpopulation; the antecedent of the new rule covers the current state, with either positive or negated predicates, and it may contain some "*don't care*" symbols.

At the end of each episode, the two standard genetic operators, mutation and crossover, are applied to the rules of the selected subpopulation. All the rules are classified in one of two sets: the set of rules that have

been tested enough (in the sense mentioned above), and those that have not yet had the chance to run enough to have a reliable strength. Only the rules belonging to this last set are subject to mutation.

The standard one-point crossover operator is applied to the *enough-tested* rules of the selected subpopulation at the end of each episode with a given probability. Both the parents are taken from the same subpopulation, and they remain in the population with a probability proportional to their strength. The children become part of the population if they are not duplicated.

The set of the enough-tested rules is in turn partitioned into three subsets, which we will describe after the introduction of the concept of steady rule. A rule r is *steady* at a given step j if its *degree of stability* $sd(r, j)$ is within a given percentage of the module of the current rule strength. The degree of stability is defined as

$$sd(r, j) = \sum_{i = ave\text{-}dev\text{-}sign}^{j} [\rho^{j-i}\delta(i)],$$

where:

- j denotes a generic episode at least *ave-dev-sign* episodes after the rule has become enough tested;
- ρ is a discount factor that gives more importance to recent $\delta(i)$;
- $\delta(i)$ is the variation of the strength s of the rule at the end of the ith episode, defined as

$$\delta(i) = |s(i) - s(i-1)|.$$

In other terms, a rule is steady when it has been tested enough, and its strength does not change too much. This is important in our application, since we have a lot of "*don't care*" symbols in the antecedents of the rules. Thus, we have general rules that may trigger in many different situations. If a rule is steady, then its strength is a good estimation of its suitability over its whole range of applicability.

The set of enough tested rules is partitioned in three subsets: the *steady rules*, the *unsteady rules*, and the *pending rules*, i.e., the rules that have been declared *enough-tested* less than *ave-dev-sign* episodes before.

A population is *steady* when a given percentage of its rules is steady. When a population is steady for a given number of episodes, the unsteady rules are eliminated from the population, and the steady rules are tested for a given number of control cycles. If their performance is high enough, they are saved and the mutation operator is applied with a given probability before continuing the learning activity.

Finally, steady and pending rules are deleted when other steady and stronger rules *cover* them, i.e., when these last could trigger in at least the

same states, with at least the same degree. An unsteady rule is deleted when there is another stronger, enough-tested rule that covers it.

5. EXPERIMENTAL RESULTS

Here, we report about two experiments done with S-ELF: in the first one, the algorithm learns a set of fuzzy metarules to accomplish a given mission; in the second one, the strategies learnt by S-ELF are used, together with other hand-coded strategies for other missions, to achieve a goal of reactive navigation. In both the cases, the reference agent was CAT, whose features exploited in the considered tasks are summarized below.

5.1. CAT, Our Reference Robot

CAT is about 60 cm long, wide, and high (see Figure 3). It has a carlike kinematics, with steering front wheels and posterior traction. It can move forward and backward at a maximum speed of 20 cm/s, and it has a maximum steering radius of 1.2 m.

For the tasks that we mention in this paper, we use data that come from sonar, bumpers, and the odometer.

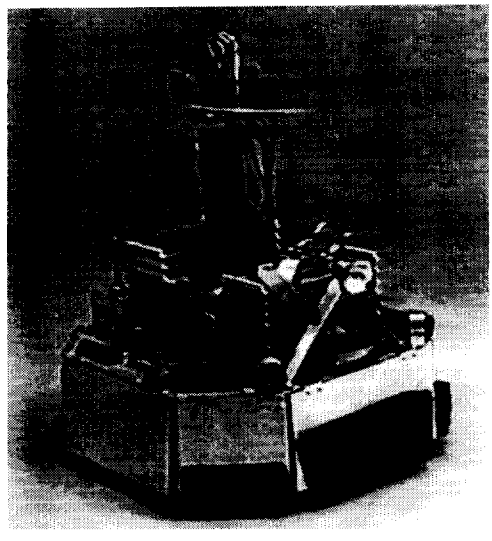

Figure 3. The robot CAT.

Sonar sensors suffer from:

- *imprecision*, since they may receive an echo from objects present in a multilobed zone in front of them;
- *low reliability* in a real environment, due to the high probability of missing echoes, and to the different reactions to different reflecting surfaces.

CAT mounts four Polaroid sonar sensors on a rotating turret, and fires them three times during a control cycle, in different positions, covering 360 degrees.

Bumpers can only detect contacts. They are very reliable.

The *odometer* is based on two free wheels, coaxial with the traction wheels and connected to two encoders. The free wheels may stick or slip from time to time on irregularities of a real floor, causing errors in the odometric measures. In the work we are presenting in this paper, data from sonar are used to limit the problems due to the intrinsic low quality of the odometer.

5.2. Learning a Rule Base

In this first experiment, we have tested the learning algorithm with reference to two different missions: *corridor following* and *aligning along a corridor*. In both cases, learning has taken place in a 3-m-wide corridor. In each learning session, more than one stable rule base has been identified; in particular, a rule base is saved when its *stability level* is higher than a *stability threshold*, whose value is continuously incremented.

- Its initial value is set to 0.5; when a rule base shows a stability higher than this for a given number of consecutive episodes, then it is saved, all the unsteady rules are eliminated, and a test phase is started.
- At the end of such a phase, the stability is higher, because of the elimination of unsteady rules; the new threshold is given by the new stability value plus an increment, equal to 0.05 in this experiment.

The test phase is composed of some *trials*, different from each other because of the initial agent position. Since such initial values are selected to cover all the space of the possible initial conditions, the agent behavior tested in this phase is considered to be a good approximation of the behavior of the agent starting from any initial position. The ability of a rule base to perform well in such trials is used to evaluate its *completeness*, defined as the sampling likelihood for the agent governed by the considered rule base to reach its goal in a given number of steps.

5.2.1. LEARNING TO FOLLOW A CORRIDOR In this first learning experiment, the agent must learn to follow a corridor that can also contain unmapped obstacles.

Table 1. Partitioning of the Rule Bases with Respect to their Completeness[a]

Completeness, %	Fraction of rule bases, %
100	97
≥ 88	98.5
≥ 77	100
< 77	0

[a]Over the total of 72 learnt rule bases.

The reinforcement program provides a positive reinforcement, proportional to the movement along the corridor and to the alignment to it, when the agent moves in the correct direction; a negative reinforcement proportional to the movement in the wrong direction is also given. Finally, a strong punishment is given whenever the agent touches a wall or an obstacle.

We have run 18 independent learning sessions. In each one of them, more than one complete rule base has been learnt: Table 1 shows the percentage of rule bases that have a completeness higher than a given value; the percentage is computed with reference to the total number of 72 rule bases, even if learnt in different sessions. In Table 2 we report the results referring to the most complete rule bases learnt in each learning session; whenever more than one had the same completeness, we have selected the rule base with the highest performance. The value reported for each rule base is obtained by averaging the values relative to the different trials during the test phase. The *performance* value is computed according to a law that gives higher performance values when the agent proceeds in the middle of the corridor *and* is oriented approximately along it. In the definition of both the performance and the reinforcement program we have adopted fuzzy models.

Table 2. Results Concerning the Best Rule Bases Learnt in Each Learning Session

Quantity	Average	Std. dev.	min	max
Number of *enough-tested* rules	75.53	12.01	49	96
Stability level	0.70	0.09	0.51	0.86
Completeness (%)	100	0	100	100
Performance	0.83	0.06	0.67	0.94
Number of moves	197.67	5.44	192.44	210.44
Number of crashes	0	0	0	0

Table 3. Results Concerning the Use of the Learnt Strategy in a 2-m-Wide[a] Corridor

Quantity	Average	Std. dev.	min	max
Completeness (%)	100	0	100	100
Performance	0.79	0.05	0.72	0.91
Number of moves	208.56	19.23	192	255.78
Number of crashes	1.19	1.63	0	5.22

[a] Instead of 3 m wide.

The first two values reported in Table 2 refer to the learning phase. The other four refer to the test phase; they are the completeness, the performance value, the number of moves to reach the end of the corridor (the minimum theoretical value is 191), and the number of contacts with walls.

We tried to use the same strategy learnt in a 3-m-wide corridor in a 2-m-wide one, obtaining the results reported in Table 3. Table 4 contains the results concerning the adoption of the same strategy in a 4-m-wide corridor. The tables show that the strategy learnt in a particular corridor is *portable*, i.e., it can be used to accomplish the task even in a different corridor. This mainly comes from the high-level description of the strategy.

Even if learning has taken place in an obstacle-free corridor, the same strategy is able to accomplish the task in the presence of obstacles, as shown in Figure 4. Remember that the *a priori* unknown obstacles are not present in the approximate map provided to the agent; therefore, it is not aware of their existence.

5.2.2. LEARNING TO ALIGN ALONG A CORRIDOR The aim of this second experiment is to shape a rule base enabling the agent to align along a corridor; this is analogous to reaching a given heading with respect to the corridor. The reinforcement program gives a reinforcement proportional to the proximity to the desired heading.

Table 4. Results Concerning the Use of the Learnt Strategy in a 4-m-Wide[a] Corridor

Quantity	Average	Std. dev.	min	max
Completeness (%)	92.16	10.61	66.7	100
Performance	0.72	0.09	0.60	0.86
Number of moves	202.98	9.42	194.11	233
Number of crashes	0.01	0.04	0	0.11

[a] Instead of 3 m wide.

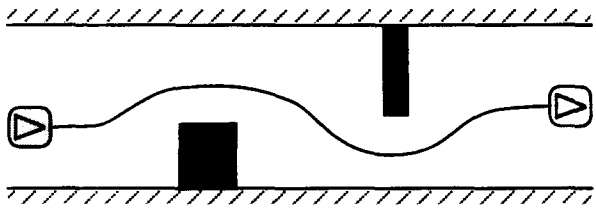

Figure 4. Example of trajectory followed by the agent in a corridor with obstacles.

We have run 18 independent learning sessions, in each of which more than one rule base was saved. Table 5, analogous to Table 1, shows the percentage of rule bases (over the total of 57 rule bases learnt in the 18 sessions) that exhibit a completeness higher than a given value.

As in the previous experiment, Table 6 shows the results concerning the most complete rule bases learnt in each learning session: the value used for each rule base is computed as an average of the values relative to the different trials of a test phase. The *performance* value is related to the improvement in agent orientation (a value equal to 1 states that the agent aligns itself along the corridor in the theoretical minimum number of moves). The first three rows of Table 6 refer to the learning phase. The other four refer to the test phase; they are the completeness, the performance value, the number of maneuvers to reach the correct orientation, and the number of contacts with walls.

We have applied the same strategy learnt in a 3-m-wide corridor in a 2-m-wide one, obtaining the results shown in Table 7. It shows that the strategy learnt in a particular corridor is able to accomplish the task even in a different corridor.

In Figure 5 we report some trajectories followed by the agent during trials. Notice how the agent can manage maneuvering also in narrow corridors.

Table 5. Partitioning of the Rule Bases with Respect to Their Completeness[a]

Completeness, %	Fraction of rule bases, %
100	87.7
≥ 75	98.2
< 77	1.8

[a]Over the total of 72 learnt rule bases.

Table 6. Results Concerning the Best Rule Bases Learnt in Each Learning Session

Quantity	Average	Std. dev.	min	max
Number of learning episodes	471.07	199.45	158	863
Number of *enough-tested* rules	46.87	6.62	38	56
Stability level	0.75	0.11	0.60	0.91
Completeness (%)	100	0	100	100
Performance	0.85	0.08	0.60	0.95
Number of maneuvers	1.65	0.21	1.42	2.28
Number of crashes	0	0	0	0

5.3. Navigation in an Approximately Known Environment

In the second group of experiments, the agent is called on to act in an environment that is known only in an approximate way. More precisely, it has an *approximate map* of such an environment that contains some *features* (corridors and doors), about which only rough and imprecise metric information are available: for example, in the map we may have a corridor about 3 m long. Approximation is represented by fuzzy values.

These experiments show that the strategies learnt by S-ELF can cooperate with some other hand-coded strategies to accomplish a complex mission, like reaching a particular target. Moreover, some of these learnt strategies come from the same mission template, showing that the set of metarules learnt in a particular environment can be successfully applied even in different environments sharing the same topological characteristics (e.g., in different corridors).

The mission definition allows us to use a *goal-regression planner* to select the missions (and so the strategies) that accomplish a complex task

Table 7. Results Concerning the Learnt Strategy Applied in a 2-m-Wide[a] Corridor

Quantity	Average	Std. dev.	min	max
Completeness (%)	91.5	11.4	55.5	100
Performance	0.82	0.09	0.68	0.99
Number of maneuvers	3.24	1.42	1.33	7.85
Number of contacts	0.19	0.18	0	0.71

[a] Instead of 3 m wide.

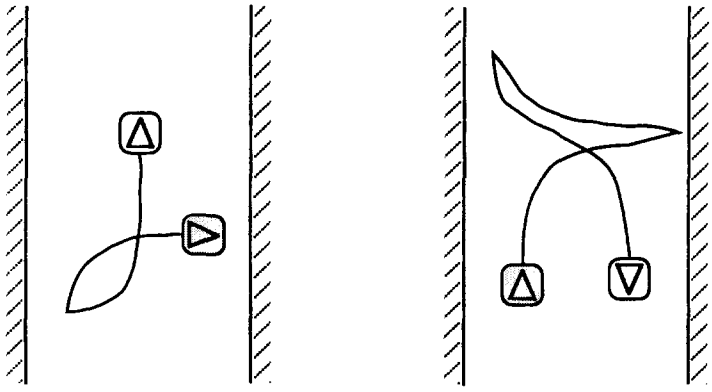

Figure 5. Examples of trajectories followed by the agent aligning along a corridor.

(*mission plan*): their activation is led, at running time, by sensorial data about the surrounding environmental conditions; in other words, it is data-driven. We have run experiments in two different conditions: without and with obstacles. It is important to note that the approximate map given to the agent contains information only about the *a priori* known elements: the obstacles present in the environmental configuration shown on the right in Figure 6 are not present in such a map. The trajectories followed by the agent in the two cases are drawn in Figure 6.

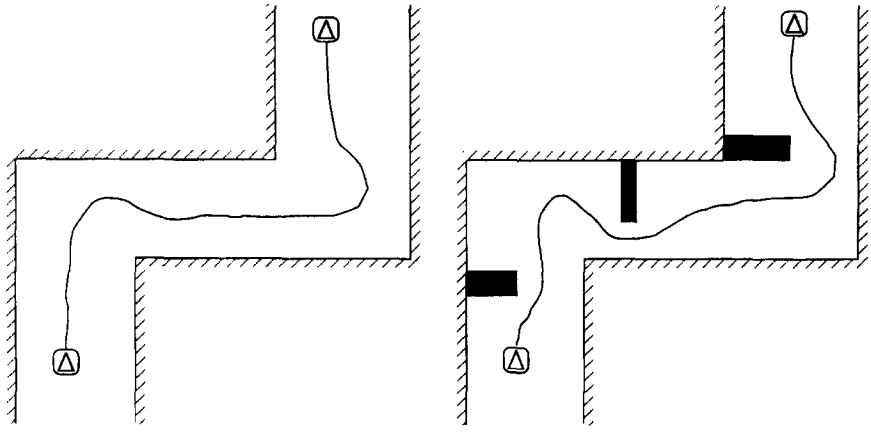

Figure 6. Examples of agent's trajectories during navigation in a complex environment.

6. CONCLUSION

We have presented an approach to support with reinforcement learning the process of development of complex behaviors for autonomous agents. S-ELF can learn in which contexts, described by fuzzy logic predicates, basic behaviors are useful. Like its ancestor (ELF), it shows many interesting features, such as robustness, flexibility, efficiency; the learnt rule bases are portable and robust.

Our current research activity is oriented towards the application of S-ELF to more complex environments and missions, and the systematic study of the limits of the portability and robustness of the learnt control systems. Moreover, we are also studying the integration of the learning and control architecture here presented with another system [7] (based on a neurofuzzy approach) that provides reliable symbol grounding [16] for the fuzzy logic predicates.

7. APPENDIX

In this appendix we report details concerning the mechanism we have devised to shape the control rules according to the truth evaluation of the context predicates.

Since the context of a rule is related to a goal-oriented viewpoint about the current agent's mission, it can be reasonably used to shape either the *desirability* of a direction of observation (through D_i) or the *utility* of a steering angle (through U_i). In fact, the *viability* V_i and the *limitations* L_i seem not to be under direct agent control, since they depend on the environment, and consequently, they cannot shape the control rules.

There are, at least theoretically, two different ways of shaping the control rules: the context can act either on the antecedents or on the consequents. It is shown in [1] and [9] that these two methods, apparently antithetical, can lead, under opportune hypotheses, to the same emergent behavior. In other words, the context can either modify the perception of the environment (S^C), or the actions on it (A^C), without any difference in the agent's interaction with the world. The two ways of shaping are referred to, respectively, as $S^C \rightarrow A$ and $S \rightarrow A^C$; in the rest of this appendix we will describe only the first one, adopted in the examples presented in the rest of the paper.

7.1. Context Shaping Desirability

In the $S^C \rightarrow A$ modality, the context shapes only the membership function of the desirability D_i.

The shape of the fuzzy set D_i is generated from the context through a generative function. In Figure 7, we show on the right the generative function $g^*(\cdot)$ that we have used to create all the D_i; on the left we show how we select the generative function itself within a family G of functions that have a triangular shape of semiamplitude δ. In other terms, we select a generative function $g^*(\cdot)$ within a family of functions having the same semiamplitude, and different coordinates for their maximum value. To understand how the selection takes place, imagine that the truth value of the context predicate [say (*in corridor-1*)] in the current situation is t^*. The function $\alpha(\cdot)$, drawn on the left of Figure 7, maps t^* to α^*; $\alpha^* = \alpha(t^*)$. As mentioned before, the generative function $g^*(\cdot)$ that belongs to the family G, and that is characterized by two specific values α^* and t^*, is unique. Thus, from the truth value of the context predicate, by using the function $\alpha(\cdot)$, we obtain a value α^* for α that, together with t^*, univocally identifies a generative function $g^*(\cdot)$. From $g^*(\cdot)$, we shape all the control rules in this way: for each $\alpha_i \in [\alpha^* - \delta, \alpha^* + \delta]$, we generate a fuzzy set $D_i = \langle \alpha_i, g^*(\alpha_i) \rangle$, whose support reduces to a single point. In our example, this represents the desirability of each heading value, given the truth value of the context predicate. In other terms, we obtain that if it is true that we are in *corridor-1*, then the desirability of having a heading α in the same direction of the corridor is maximum, and it has a shape like the one presented in Figure 7, determined by $g^*(\cdot)$.

Given that the context (in our case, being in *corridor-1*) cannot influence the viability of a direction, we can say that $S_i = D_i$.

Now, still focusing on this example, it is evident that there is a direct mapping between the state and the control variable: for each heading there is one and only one steering angle allowing the agent to move in that direction. Thus, a control rule

$$IF\ (s\ IS\ S)\ THEN\ (a\ IS\ A)$$

is created, by introducing the corresponding $A_i = \langle f(\alpha_i), L(f(\alpha_i)) \rangle$, where $f(\cdot)$ is the function that represents the above-mentioned mapping between

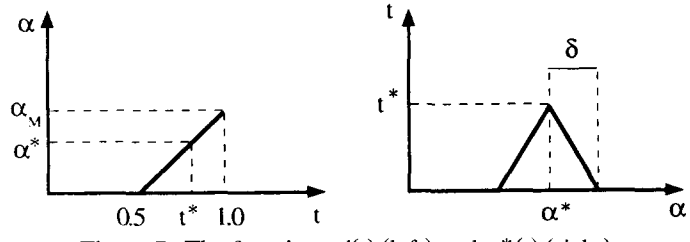

Figure 7. The functions $d(\cdot)$ (left) and $g^*(\cdot)$ (right).

state variable (e.g., direction of observation) and control action (e.g., steering angle). Note that U_i does not appear in the definition of A_i: this is because we have decided, in this modality, that the context influences only the antecedent part of a control rule.

References

1. Basso, F., S-ELF, a learning system for the coordination of fuzzy behaviors for an autonomous robot (in Italian), Master's Thesis Politecnico di Milano, Milan, Italy, 1996.
2. Bonarini, A., ELF: Learning incomplete fuzzy rule sets for an autonomous robot, *Proceedings of EUFIT '93*, ELITE Foundation, Aachen, Germany, 69–75, 1993.
3. Bonarini, A., Learning to coordinate fuzzy behaviors for autonomous agents, *Proceedings of EUFIT '94*, ELITE Foundation, Aachen, Germany, 475–479, 1994.
4. Bonarini, A., Evolutionary learning of fuzzy rules: Competition and cooperation, in *Fuzzy Modeling: Paradigms and Practice* (W. Pedrycz, Ed.), Kluwer Academic, Norwell, Mass., 265–284, 1996.
5. Bonarini, A., Delayed reinforcement, fuzzy Q-learning and fuzzy logic controllers, in *Genetic Algorithms and Soft Computing* (F. Herrera and J. L. Verdegay, Eds.), Physica Verlag (Springer-Verlag), Heidelberg, 447–466, 1996.
6. Bonarini, A., Learning dynamic fuzzy behaviors from easy missions, *Proceedings of the IPMU '96*, Proyecto Sur de Ediciones, Granada, 1223–1228, 1996.
7. Bonarini, A., Symbol grounding and a neuro-fuzzy architecture for multisensor fusion, *Proceedings WAC-ISRAM*, TSI Press, Albuquerque, N.M., to appear.
8. Bonarini, A., Anytime learning and adaptation of structured fuzzy behaviors, *Adaptive Behavior J.*, to appear.
9. Bonarini, A., and Basso, F., Designing fuzzy logic behaviors, to appear.
10. Booker, L. B., Classifier systems that learn their internal models, *Machine Learning* 3, 161–192, 1988.
11. Brooks, R. A., A robust layered control system for a mobile robot, *IEEE Trans. Robotics and Automation* 2(1), 14–23, 1986.
12. Brooks, R. A., Intelligence without representation, *Artificial Intelligence* 47, 139–159, 1991.
13. Dorigo, M., and Colombetti, M., Robot shaping: Developing autonomous agents through learning, *Artificial Intelligence* 71, 321–370, 1995.
14. Dubois, D., and Prade, H., *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, Orlando, Fla., 1980.

15. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.

16. Harnad, S., The symbol grounding problem, *Phys. D* 42, 335–346, 1990.

17. Kaelbling, L. P., Littman, M. L., and Moore, A. W., Reinforcement learning: A survey, *J. Artificial Intelligence Res.* 4, 237–285, 1996.

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

19. Mataric, M. J., A distributed model for mobile robot environment—learning and navigation. Tech. Report 1228, MIT AI Lab., Cambridge, Mass., 1990.

20. Mataric, M. J., Integration of representation into goal-driven behavior-based robots, *IEEE Trans. Robotics and Automation* 8(3), 304–312, 1992.

21. Saffiotti, A., Konolige, K., and Ruspini, E. H., A multivalued logic approach to integrating planning and control, *Artificial Intelligence* 76(1–2), 481–526, 1995.

22. Watkins, C., and Dayan, P., Q-learning, *Machine Learning* 8, 279–292, 1992.

23. Wilson, S. W., Classifier fitness based on accuracy, *Evolutionary Comput.* 3(2), 149–175, 1995.