Learning context motivation in coordinated behaviors

Andrea Bonarini, Matteo Matteucci *

Abstract. We present a novel architecture to implement fuzzy behaviors for autonomous robots and an approach to adapt them to a dynamical environment, in real–time. We adopt the system that we have implemented in different applications such as robotic soccer (Robocup), document delivery in office environments, and surveillance.

Keywords. *Behavior-based Robotics, Reinforcement Learning, Fuzzy Systems.*

1 Introduction

We propose a novel behavior architecture, based upon fuzzy models to implement reactive autonomous agents that execute real–time tasks in dynamic environments. We propose fuzzy models to face the issue of uncertain perception; a fuzzy model implements a classification of the available information and knowledge in terms of fuzzy predicates, which have been demonstrated to be a powerful and robust representation paradigm [9][13].

Facing real–time tasks in dynamic environments, we have to overcome the time constrains deriving from a purely symbolic planning using a reactive approach based on the interaction of several simple specialized behaviors, performing local reasoning. We associate to reactive behaviors a context evaluation mechanism to tune the behaviors to specific situations.

The main purpose of this paper is to present a behavior architecture suitable for learning coordinated behaviors. We also describe an approach to learn context driven coordination among basic behaviors. We have designed such an architecture to accomplish the learning needs. In our *learning oriented* behavior architecture, a behavior module is associated with two sets of fuzzy predicates: the *CANDO enabling preconditions* and the *WANT context conditions*. The truth values of CANDO predicates enable the execution of the associated behavior and the truth values of WANT predicates weight the actions proposed by the behavior module considering the activation context. The output of the behavior modules are actions to be executed by the robot's physical actuators or messages to be sent to other agents.

We adopt this architecture on different robots involved in a wide variety of tasks, such as: soccer playing in Robocup [2], document delivery in an office environment, and surveillance.

We believe that learning may support the development and adaptation of robotic agents. In particular, we treat in the second part of this paper aspects related to

*This research has been partially supported by the MURST Project CERTAMEN. The authors are with the AI and Robotics Project, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci, 32, I-20133 Milano, Italy - E-mail: {bonarini, matteucc}@elet.polimi.it

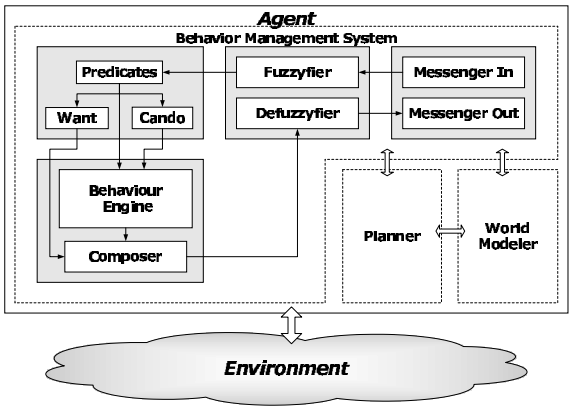

Figure 1: The agent architecture

adaptation in a short time to new environments by tuning the context predicates associated to coded (or previously learned) basic behaviors. We present an approach to this task which is feasible also in applications such as those mentioned above, where the environment is dynamic and subject to drastic changes.

2 Behavior architecture

Our overall architecture (see figure 1) consists of a behavior management system interfaced with other subsystems such as: a *world modeler* and a *planner*. The former maintains a model of the world interfacing with the available sensors and other, eventually present, collaborating agents; its main functions are sensor fusion and self-localization. The planner states the goals for the agent, implementing an abstract, global reasoning: the obtained plan includes information useful to evaluate the actual context. In figure the *actuation module* is not included; it is in charge of realizing the commands issued by the behavior manager by interfacing the agent with the environment. In this paper, we focus only on the *behavior management system* shown in details in figure 2.

The characterizing aspects of the behavior management system are the input interface based on fuzzy predicates and the possibility to enable or inhibit behavior modules and to graduate their proposed output. Fuzzy predicates are a general and robust [9] modeling paradigm, close to the designer's mental models [18]. Other researchers [13][10] have adopted them to implement control systems for autonomous robots for analogous reasons. Fuzzy predicates may represent aspects of the world, goals, and information coming from other agents. Their general shape consists of a fuzzy variable name, a label corresponding to a fuzzy set defined on the range of the variable, a degree of matching of the corresponding data to the mentioned fuzzy set, and a reliability value to take into account the quality of the data source. For instance, we may have a predicate represented as

$$< BallDistance, VeryClose, 0.8, 0.9 >$$

which can be expressed in natural language as: "the ball is considered very close, with a truth value of 0.8 (coming from the fuzzyfication of the incoming data, namely the real-valued distance from the ball), and a reliability value of 0.9, qualifying the data as highly reliable.

We consider ground and complex fuzzy predicates. *Ground fuzzy predicates* range on data available to the agent, and have a truth value corresponding to the degree

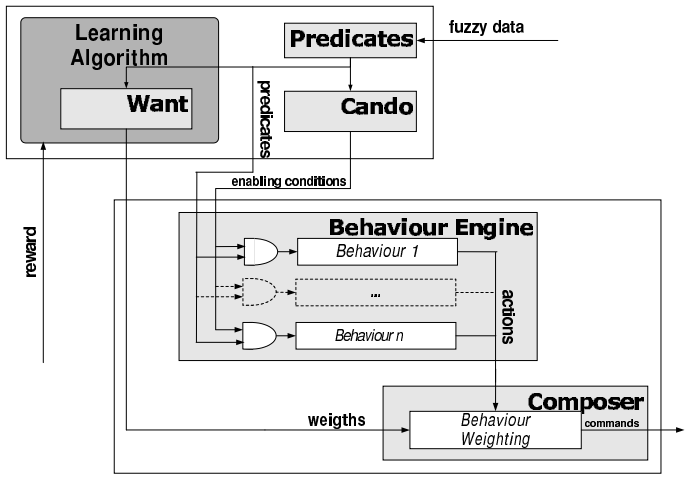

Figure 2: The behaviour management system

of membership of the incoming data to a labeled fuzzy set. This is equivalent to classify the incoming data into categories defined by the fuzzy sets, and to assign to this classification a weight between 0 and 1. Fuzzy ground predicates are defined on: sensing, present goals, or messages coming from other agents; the reliability of data is provided, respectively, by the world modeler (basing on local considerations), the planner stating the goals, and a special purpose module that computes the reliability from the other agent's reliability and the reliability it has associated with data in its message. For instance, in Robocup, teammates provide their position on the playground, together with the estimation of the quality (reliability) of their present self-localization. This information is matched against the internal model the agent has of the situation, and this provides another contribution to data reliability. Moreover, the agent maintains information about the reliability of its teammates. All this reliability information is composed to update the internal model.

A *complex fuzzy predicate* is a composition (obtained by fuzzy logic operators) of fuzzy predicates. Complex fuzzy predicates extend the basic information contained in ground predicates into a more abstract model. In RoboCup, for instance, we can model the concept of ball possession by the $HaveBall$ predicate, joining by the AND operator the predicates $< BallDistance, VeryClose >$, ground predicate deriving from the fuzzyfication of the ball distance perception, and $< BallDirection, Front >$, ground predicate deriving from the fuzzyfication of the perception of ball direction.

We define for each behavior module a set of predicates that enable its activation: the *CANDO preconditions*. The designer has to put in this set all the conditions which have to be true, at least to a significant extent, to give sense to the behavior activation. For instance, in order to consider to kick a ball into the opponent goal, the agent should have the ball control. It is quite easy for the designer to define such sets of predicates, and we do not consider useful to learn them.

Another set of fuzzy predicates is associated to each behavior module: the *WANT context conditions*. These are predicates that represent the motivation for the agent to activate a behavior in relation with a context. They may come either from the environmental context (e.g., "there is an opponent in front of me"), or from internal goals (e.g., "I have to score a goal"), or from information directly coming from other agents(e.g., "I'm taking care of the ball"). All these predicates are composed by fuzzy operators, and contribute to compute a motivational state for each behavior. So the

agent knows that it could play a set of behaviors (those enabled by the CANDO preconditions), but it has to select among them the behaviors consistent with its present motivations. If more than one behavior is in this condition, the actions proposed by the selected behaviors are composed with the weights computed by each behavior. This is coherent with some early work on this topic [1] [13] [10], although we expect that, in the large majority of situations, a single behavior will be activated at a time, since the composition of different behaviors is usually less effective. This is achieved through a specialization of the context model for each behavior. We have decided to spend on-line computational resources to learn the relationship between the WANT context conditions and the behaviors, as described in the next section.

With respect to the Brooks's [7] behavior architecture we introduce many differences; our behaviors management system works on fuzzy predicates obtaining the capability of coordinating the concurrent execution of several behaviors. We consider the adoption of CANDO and WANT predicates as an alternative choice to the implementation of the subsumption architecture. It is more general than Brook's proposal and also more effective in strongly dynamic environments. In our implementation the enabling connections among behaviors are context dependent (using the WANT conditions), so, in our architecture, relationships among behaviors are not rigidly defined, but we can adapt the emerging global behavior, depending also on external conditions and motivations.

Another reference we have considered is Arkin's schema-based behavior architecture [1]. It is possible to map the basic principles of our and Arkin's approaches into each other. The main difference is the fuzzy model we put at the basis of our architecture, whereas the same concepts are represented by Arkin with different tools such as a *gain* value to weight each behavior contribution, or their representation in terms of potential fields. A fuzzy model provides a powerful and more uniform representation of analogous features. A difference between our architecture and both Brooks' and Arkin's is the presence in ours of a world modeler that interfaces the external world with the behavior module. In the mentioned architectures, world modeling is embedded in each behavior definition. We have implemented such a module to achieve efficiency and to provide a unified interface from the environment to the behaviors.

In our architecture we start from Saffiotti's approach [13][10]to the use of fuzzy logic in robotics [12], which tries to face the problem of designing an effective controller for mobile robots by combining goal-specific strategies by resolution of conflicts between multiple objectives. We keep separate the CANDO from the WANT conditions, due to they different semantics, while in [13] they are put together in the *desirability* parameter.

3 Learning and adaptation

In this paper we also present an algorithm belonging to the *anytime learning* [8] class of algorithms, able to provide a result at anytime during its activity. This is important when the robotic agent has to learn while performing. In our case we use this algorithm to adapt the motivations of the agent to a dynamic environment. At the beginning, the agent interacts with the environment by selecting behaviors with the motivation model already learnt till that moment. As it verifies its effectiveness it may modify the model to try to improve its performance.

We adopt a reinforcement learning [16] schema. When the agent has to select a behavior, we consider the set of all the behaviors whose CANDO preconditions match the situation above a given threshold (*Triggerable Behavior Set - TBS*), and the set of predicates which match the situation. Some of them already contribute to the mo-

tivations of the behaviors in TBS, the others may become part of them. We add to the motivations some of the predicates not yet included. We select these predicates by considering the trade off between exploration and exploitation, the degree of matching of the candidate predicates, and their reliability. We combine the proposed actions weighting them by the corresponding motivation value. Usually, the so-produced action modifies the environment, or, at least, its perception by the agent. If the modification is relevant, it is evaluated by a reinforcement function. The relevance of the reached situation depends on the application and is defined a priori together with the reinforcement function. For instance, in the Robocup environment, we take as relevant events such as: ball possession change, goal scoring, and fouls. It is important to have reinforcement at a frequency that enables to have enough information from the environment (too scarce reinforcement negatively affects adaptation) and to allow enough time to evaluate a set of behavior motivations. For instance, if we had reinforcement at each control step, probably we cannot filter out drops in reinforcement due to accidental causes. This principle has been deeply discussed in [3][4]. We keep the same set of WANT conditions for a behavior until the sum of their matching degrees in the past does not exceed a given threshold which is an estimation of the exploitation needed to reliably evaluate the set of conditions. The obtained reinforcement is distributed to the relationship between the set of the current motivation predicates and the triggered behavior, according to the formula:

$$V_t(p,b) = V_{t-1}(p,b) + \alpha\xi(r_t - V_{t-1}(p,b))$$

where $V_t(p,b)$ is the value of the relationship between the set of predicates p and the behavior b at time t, α is the learning rate, r_t is the reinforcement at time t and ξ gives account of the fuzzyness of the model. It is proportional to the conjunction of the matching degrees of the involved predicates, computed by a T-norm [9] such as *min*. This means that the contribution coming from the current reinforcement is proportional to the degree of matching of the selected predicates. By modulating α, we can tune the system to adapt to frequent changes (high α), or to learn more foundational aspects (low α). This choice is done off-line, by considering the dynamics of the environment and the time available to learn new models. For instance, in the Robocup application we keep α high, since we do not have much learning time (6000 steps for 10 minutes); moreover the actions where the agent is involved do not span on all the available time, and we focus on the identification of the best overall behavior for a given team and match. In the surveillance task, the agent has to adapt the behavior to a new environment which is supposed to remain the same for the rest of the operation, so α can be high. In the document delivery task, the agent has to adapt to the presence of different persons working in the same environment, which may interact with the agent in different ways. So we have to filter out these too fast changes, and learn only the basic aspects with a low α.

Apart from the ξ term, the reinforcement distribution formula is classical. We share with other researchers [14] the opinion that this formula is better than more common formulas in reinforcement learning, which consider also a term taking into account the expected future reinforcement, such as, for instance, *Q-learning* [17] or $TD(\lambda)$ [15]. Our choice makes sense in environments, like those we are considering, where future reinforcement depends in large part from actions done by other agents, which give high variability.

The highest–valued motivation set is always kept for each behavior and it is compared with each new motivation set as this is considered to have been enough tested.

When the comparison is done, if the new motivation is higher–valued, it is kept, and the adaptation process continues from it, otherwise, new possibilities are explored from the old WANT conditions set. When a motivation set becomes too large, it is reduced by dropping a condition among those which induced the smallest increment in performance when they where introduced. The new motivation set is tested and evaluated as any other.

From what reported above it is possible to understand how, by manipulating on–line the context predicates for each behavior module, it is possible to adapt the cooperation among them to the needs of a specific environment. Moreover, since we include in the predicates also information about other agents, this is also a mechanism to adapt a cooperative behavior among different agents, as we are doing in Robocup, and we plan to do in the other mentioned tasks. Notice that this model may support agent cooperation based either on explicit communication (we interpret messages as predicates), or on implicit communication obtained by observing and producing actions in the environment. We are experimenting both the approaches with the same architecture introduced in this paper.

4 Experimental results

Here, we present results obtained in simulation in the Robocup environment. Simulation is a key approach in this application, since it gives the possibility to design and implement in a short time behaviors and mechanical features of opponents, making it possible to perform many different trials before playing matches against real opponents. Thus, it is also possible to identify a number of different initial configurations for the behavior system, to be used as a starting point for the actual matches.

We are involved in the F-2000 Robocup league [2] with the Italian National Team (ART - Azzurra Robot Team [11]), with our player *Rullit* [6]. We have implemented a simulator for F-2000 robots enough detailed to be a realistic tool to test strategic aspects of the game. In particular, we may attach to the simulator the same control modules implemented for the actual robot and test their behavior without needing to have two teams actually present on a real field. This makes it possible to perform realistic experiments also before the real competitions.

In the experiments about adaptation that we present here, we had the goal to adapt the control system for one player of a team that have to effectively face other teams showing predefined, canonical behaviors, similar to those seen on the field during the last World Championship. This is a realistic situation, since we will probably put on the field only one player with the ART national team, and we will have to interact with the teammates that do not have any adaptation facility on board. We have implemented strategies for the opponents such as: "play on zone", where each robot actively operates only in a predefined zone of the field and roles are defined by the assigned zone, and "all on the ball", where all the robots tend to go on the ball, thus creating a sort of barrier around it. Every 100 msec the control system takes information from the environment and computes the action to be done. The duration of the match is 20 minutes, divided in two half times. We have fed our robot with the control system adopted for Rullit in the last World Championship, and let the above described adaptation algorithm to work.

The reinforcement function gives a positive reinforcement when our team gain the possess of the ball and when we score a goal. A negative reinforcement is given when the opponents gain the ball and score a goal, and also when the adapting agent makes

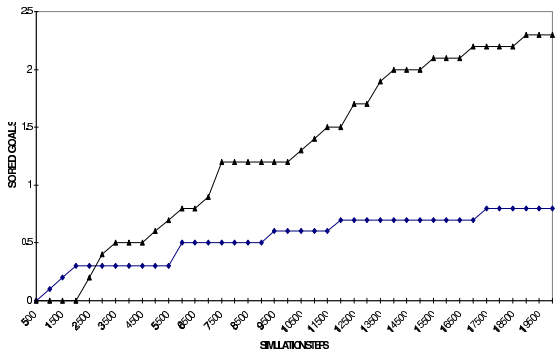

Figure 3: The performance of the team with an adapted agent

a foul. All the reinforcement values are multiplied by a given factor β in $[0,\ 5]$, when the adapting agent is directly involved in the corresponding action. We have performed many experiments, varying β, α, and the strategies of both teams. In all the experiments we have obtained interesting modifications of the original behavior, with a significant improvement in performance, for the whole team, thus demonstrating that, even with only one adapting agent in a team, the global performance increases. As an example, in figure 3 we show the number of goals scored by our team after the adaptation of Rullit with $\alpha = 0.1$, $\beta = 5$, and both the teams following the "all on the ball" strategy. The plot shows results averaged over 12 different experiments. The test trial for each experiment is done for 20000 simulation steps (about half a hour game). The lower plot is the performance of the team without adapting agents, the higher one contains the adapted *Rullit*. The quality of results is evident.

5 Conclusion

We have presented in this paper a behavioral architecture for an autonomous robot based on a fuzzy model. It has been designed to make it possible the interaction among different behavior modules, and to put in evidence aspects that can be learnt. In particular, we are presently adopting it in different environments, for different tasks, adapting the behavior of single or cooperating robots to dynamical environments in real time. We operate on-line on the fuzzy predicates that implement the context model for each behavior activation, which provide the agent with a motivation to activate the behavior itself. The fuzzy model gives robustness to the system with respect to imprecision in data acquisition and uncertainty. Moreover, it enables smooth transitions between different behaviors, a property highly desirable in learning and adaptation [5]. The presented adaptation algorithm can be considered an anytime algorithm, since at any moment it provides a sub-optimal solution to the problem, and the best solution so far found can be retrieved at any time. We have demonstrated in simulation its effectiveness in hard tasks such as Robocup.

References

[1] R. C. Arkin. *Behavior-Based Robotics.* MIT Press, Cambridge, MA, 1998.

[2] M. Asada, H. Kitano, I. Noda, and M. Veloso. Robocup: today and tomorrow – what we have learned. *Artificial Intelligence Journal*, 110:193–214, 1999.

[3] A. Bonarini. Evolutionary learning of fuzzy rules: competition and cooperation. In W. Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*, pages 265–284. Kluwer Academic Press, Norwell, MA, 1996.

[4] A. Bonarini. Anytime learning and adaptation of hierarchical fuzzy logic behaviors. *Adaptive Behavior Journal*, 5(3–4):281–315, 1997.

[5] A. Bonarini. Reinforcement distribution to fuzzy classifiers: a methodology to extend crisp algorithms. In *IEEE International Conference on Evolutionary Computation – WCCI-ICEC'98*, volume 1, pages 51–56, Piscataway, NJ, 1998. IEEE Computer Press.

[6] A. Bonarini. The body, the mind or the eye, first? In *Proceedings of the Third International Workshop on Robocup (Robocup99)*, pages 40–45, Cambridge, MA, 1999. IJCAI Press.

[7] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robot Automation*, 2(1), 1986.

[8] J. J. Grefenstette and J. Ramsey. An approach to anytime learning. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Conference on Machine Learning*, pages 189–195, San Mateo, CA, 1992. Morgan Kaufmann.

[9] G. J. Klir, B. Yuan, and U. St. Clair. *Fuzzy set theory: foundations and applicatons*. Prentice-Hall, Upper Saddle River, NY, 1997.

[10] K. Konolige, K. Myers, E. Ruspini, and A. Saffiotti. The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):215–235, 1997.

[11] D. Nardi, G. Adorni, A. Bonarini, A. Chella, G. Clemente, E. Pagello, and M. Piaggio. Art – azzurra robot team. In M. Asada M. Veloso, E. Pagello, editor, *RoboCup 98–Robot Soccer World Cup III*, Berlin, D, 2000. Springer Verlag.

[12] A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(1):180–197, 1997.

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

[14] P. Stone and M. Veloso. Tpot-rl: Team-partitioned, opaque-transition reinforcement learning. In M. Asada, editor, *RoboCup 98: Robot Soccer World Cup II*, pages 221 – 236, Berlin, D, 1998. Springer Verlag.

[15] R. S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[16] R. S. Sutton and A. G. Barto. *Reinforcement Learning An Introduction*. MIT Press, Cambridge, Massachusetts, 1999.

[17] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[18] L. A. Zadeh. Making computer think like people. *IEEE Spectrum*, pages 26–32, 1984.