A novel model to rule behavior interaction

Andrea Bonarini, Matteo Matteucci, Marcello Restelli *

Abstract. We present a novel model for the management of behavior interaction in behavior-based systems. Most of the approaches so far presented require to have knowledge about the different behavior to manage effectively the interaction among the actions each behavior suggests. We propose a model based on *hierarchical informed behavior composition*, where each behavior is informed about the actions proposed by other – less critical – behaviors, and it can take decisions that take into consideration these proposals. Thus, the high-level control system can be really designed in a modular way, and the interaction among behaviors can be easily designed and optimized. We have implemented this architecture, and we have used it in Robocup competitions and service robotics.

Keywords. *Behavior-based Robotics, fuzzy behavior architecture, hierarchical informed behavior composition.*

1 Introduction

Several autonomous robotic applications are developed according to the behavior-based paradigm (e.g., [2] [6]). In principle, a behavior is a simple functional unit that takes care only of the achievement of an elementary goal, on the basis of a small subset of the information from the input space. The behavior-based approach allows to design complex robot controllers in a modular way, by a suited combination of different behaviors. Behaviors should be designed independently from each other according to the *independent design principle*. The main issue of this approach is that, in large applications, having a large number of heterogeneous goals, it may be difficult to design the correspondingly large number of behaviors so that their interaction brings the desired results. The crucial point is: how to combine the output of many behaviors, by preserving the independent design principle, and still obtain predictable results? Researchers have proposed several behavior coordination mechanisms [8], in order to get a good behavior interaction, but each of them shows potential problems, summarized here below.

The subsumption architecture [6] consists of a set of behaviors, organized in carefully crafted networks that specify behavior interactions. Behaviors that pursue the same goal are grouped at the same level. The higher-level behaviors may inhibit the output of lower-level behaviors. This calls for knowledge about the relationships among behaviors, in contrast with the principle of independent design.

Fuzzy rule based systems have gained a good success in mobile robotics. In these systems, the action selection process is carried out by using fuzzy logic and inferencing. Saffiotti, Ruspini, and Konolige [9] have been among the first researchers working on

*The authors are with the AI and Robotics Lab, Department of Electronics and Information, Politecnico di Milano, Piazza Leonardo da Vinci, 32, I-20133 Milano, Italy - E-mail: {bonarini, matteucc, restelli}@elet.polimi.it

this topic; they have proposed to introduce a coordination mechanism called *context-dependent blending*, which allow to merge the actions proposed by behaviors with different weights according to the current situation. This leads to nice, smooth behaviors, but there are cases where blending does not provide a suitable action. For instance, when facing an obstacle, blending the actions proposed by two behaviors suggesting to go, respectively, on the left and on the right may produce a resulting action that brings the robot straight against the obstacle. To solve this kind of problems, behaviors should be designed carefully, eventually considering the possible blending with other behaviors, and this is again in contrast with the principle of independent design.

Also Arkin [1] uses a behavior-based approach; he decomposes the control problem into a set of schemata which relate sensor input to motor output. In his architecture, all schema outputs are merged, with output gains being used to implement a priority mechanism. Again, knowledge about other behaviors (their gains) is needed to produce a robot behaving as expected.

To solve some of the above mentioned problems, we have implemented Mr.BRIAN (Multilevel Ruling BRIAN), an innovative extension to our previous architecture BRIAN (BRIAN Reacts by Inferring ActioNs) [3]. BRIAN substantially followed the lines traced by Saffiotti et al. [9], by composing actions according to the value of fuzzy predicates matching the context description. In order to reduce the design complexity and to increase the modularity of our approach, we propose to organize behaviors into a hierarchical structure. Behaviors are placed in the hierarchy according to their *priority*. Each behavior reacts not only to context information, but also to actions that potentially interfere with its goals, and that may have been proposed by the lower-level (less critical) behaviors. In this way, a behavior knows what the other lower-level behaviors would like to do, and it can try to achieve its goal while trying to preserve, as much as possible, the actions proposed by others. The flow of this kind of information allows behaviors to implicitly communicate their goals to higher-level behaviors. This interface makes easier to build behaviors in a modular way, since each behavior can be unaware of what the other behaviors are designed for, and of which are the behaviors that are proposing actions. The actions taken into consideration when designing a behavior are those that could potentially interact with the actions that the behavior would propose to achieve its goals. There is no assumption about the possibility that these actions are actually proposed by some other behavior. The designer is free to add or remove behaviors without changing nothing else in the system. Whenever a behavior receives as input proposed actions that are in contrast with its own goal, the behavior can inhibit the received actions and propose others. This implies that behaviors at the higher levels have higher priority, so that they can impose their wishes by overriding the actions proposed by others.

Our hierarchical organization of behaviors is different from that presented in other related works. In [11], the authors define two classes of behaviors: the *primitive* behaviors, that are simple reactive behaviors that operate at the lowest level and implement distinct fuzzy control policies, and the *composite* behaviors that, acting as a fuzzy decision system, coordinate primitive behaviors in order to achieve higher-level goals. These last are not behaviors in the original sense, and implement a sort of higher level control: the problem of behavior blending is left unsolved, since governing it by composite behaviors would mean to forget the independent implementation principle. In [7], each behavior is composed of two parts: the *Action* and the *Activity* sub-modules. The Action sub-module produces the commands for the actuators, while the Activity sub-module produces a value that is associated to the priority of the behavior. Also in this

architecture, the hierarchical approach is used to specify behavior sequences by defining *coordinator* behaviors, which can enable or disable underling behaviors. Recently, Saffiotti and Wasik [10] have defined a hierarchical fuzzy behavior architecture, in which complex behaviors are built by combining simpler behaviors through the use of fuzzy meta-rules, in order to modulate the activation of a behavior according to the context. The command fusion is realized through standard fuzzy fusion techniques. Also this system suffers for the above mentioned problems.

In our architecture the hierarchical approach has not been introduced to solve the *arbitration problem* (i.e., which behavior should be activated at any moment), as done in the mentioned implementations, but it deals with the *command fusion problem* (i.e., how to combine the actions proposed by different behaviors into one command). Our hierarchy is made up only of primitive behaviors, which can solve possible conflicts by reasoning on actions potentially proposed by others. In our architecture, the arbitration problem is faced by planning and coordination activities, carried out by other modules (see [5]).

The following section presents Mr.BRIAN in details, with some examples that highlight the advantages of our approach. Section 3 describes the practical application of Mr.BRIAN in the RoboCup domain. We conclude with a discussion of results and directions for future activity.

2 The interaction model in Mr.BRIAN

Mr.BRIAN is aimed at reducing the complexity of designing a behavior-based robot control system, and to increase the modularity/reusability of behavioral modules.

In our previous system BRIAN [3] we have faced the issue of controlling the interactions among behaviors by decoupling them using context conditions described in terms of internal state, environmental situation, goals, and coordination with other agents. This was done using two sets of fuzzy predicates associated to each behavioral module: *CANDO* and *WANT* conditions. CANDO conditions are used to decide if the behavioral module is appropriate to the specific situation: if they are not verified, the behavior activation does not make sense. 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. WANT conditions are predicates that represent the motivation for an agent to execute a behavior in relation with the actual context. Composition of behavior modules active at the same time is implemented by these WANT conditions that semantically represent the opportunity of executing the action proposed by a behavioral module in a given context. They may come either from the environmental context (e.g., *TargetInFront*, which may be expressed as "the target is in front of me"), or from strategic goals (e.g., *CollectDocuments*, "I have to collect the documents to be delivered").

In Mr.BRIAN, shown schematically in Figure 1, behaviors are organized in a hierarchy according to their *priority*; each behavior receives as input also the actions proposed by the lower-priority behaviors. In this way, higher-priority behaviors know lower-priority behavior intentions and they can try to achieve their goals while preserving, as much as possible, actions proposed by others. We call this approach to behavior coordination *Informed Hierarchical Composition* since we have a flow of information regarding proposed actions from lower-priority to higher-priority behaviors that allows for an implicit communication of goals to higher-priority behaviors. This approach makes easier to build behaviors in a modular way, since each behavior doesn't have to

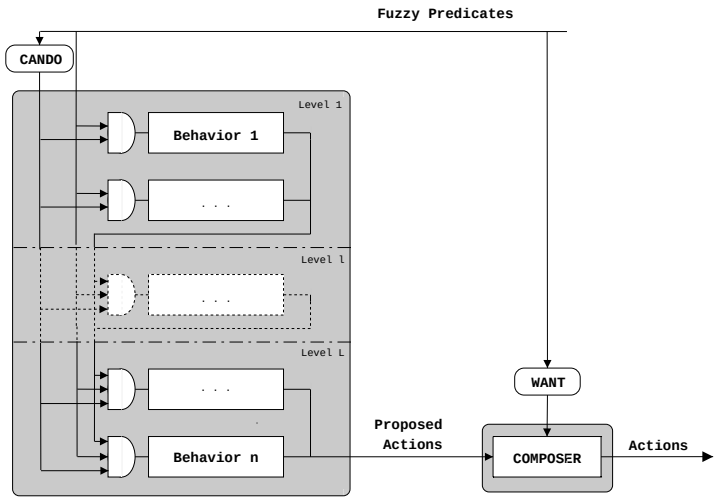

Figure 1: Multilevel Ruling BRIAN

be aware of what the other behaviors are designed for, but can reason on their proposed actions. With this approach, the designer is free to add or remove behaviors without changing anything else in the system and whenever a behavior receives as input proposed actions that are in contrast with its own goal it can drop them and propose new ones. This requires only the designer to put in the higher levels behaviors that have higher priority, so that they can impose their wishes by overriding the action proposed by the other behaviors.

To better understand Mr.BRIAN interaction model, consider the i^{th} behavioral module as a mapping from input variables to output variables implemented as a fuzzy logic controller (FLC) where a set of fuzzy rules matches a description of the situation given in terms of fuzzy predicates, and produces predicates representing actions a_i^k with associated a desirability value $\mu_{a_i^k}$ obtained from the matching degree of rule preconditions with the actual situation.

We associate a level number to each behavioral module and any module B_i^l at level l receives as input, besides the sensorial data, also the actions proposed by the modules which belong to the level $l-1$. Behaviors (except those at level 0) can predicate on the actions proposed by modules at lower levels, discard some actions, and propose other actions. Let $A = \bigcup_k a^k$ the set of all possible actions, we have for the i^{th} behavior belonging to the l^{th} level

$$B_i^l : I_i \times A^{l-1} \mapsto A_i^l \cup \overline{A}_i^l,$$

where $A^{l-1} \subset A$ is the set of the actions proposed by all the modules at level $l-1$, $A_i^l \subset A$ is the set of the action proposed by the behavior, while $\overline{A_i^l} \subset A$ is the set of actions that should be removed. Thus, given $A^0 = \emptyset$, the actions proposed by the generic level l can be expressed by the following formula:

$$A^l : \left(A^{l-1} \setminus \bigcup_i \overline{A}_i^l \right) \cup \bigcup_i A^l.$$

In the Mr.BRIAN perception-action loop, we first compute the CANDO conditions and select the behaviors that can be activated, since they have a CANDO value μ_i^C higher than a given threshold τ, which can be defined by the designer or even learnt by experience [4]. Then, Mr.BRIAN computes the A_i^l produced by the selected behaviors associating each of them with the respective μ_i^C value. Finally, the motivation conditions

are evaluated, and the result μ_i^W is used to weight the action desirability for each behavior using $\mu_i^W \cdot \mu_{a_i}^k$. The final actions a^K comes from the weighted average of proposed action with these weighting values:

$$a^K = \frac{\sum_{a^k} \max_i(\mu_i^W \cdot \mu_{a_i}^k) \cdot a^k}{\sum_{a^k} \max_i(\mu_i^W \cdot \mu_{a_i}^k)}.$$

In Mr.BRIAN, we use CANDO and WANT conditions to allow the designer to create a dynamic network of behavior modules defined thought context predicates. With respect to usual hierarchical behavior-based architectures (e.g., subsumption architecture) we do not have a complex predefined interaction schema that has to take into account all possible execution contexts. In fact, at any instant, the agent knows that it could play a limited set of behaviors (i.e., those enabled by the CANDO conditions), and it has just to select among them the behaviors consistent with its present WANT motivations.

3 Mr.BRIAN in the RoboCup domain

RoboCup is an international robot soccer competition, organized in several leagues, which differ in the type of robots and environment involved. We used Mr.BRIAN for the high-level controller of robots that participated to the middle-size league. In this league each team is made up of four robots: a goalkeeper and three players. All the robots are fully autonomous and can only use onboard sensors. Communication among robots is allowed.

All the robots in out team, except the goalkeeper, use the same behavior hierarchy (see Figure 2), where we have organized our modules on five levels. At the first level, we have put those behaviors whose activation is determined also by the information coming from the planning layer. At this level, we find both purely reactive modules and parametric modules (the latter can be distinguished by a couple of round brackets after their name). The higher levels contain only purely reactive behavioral modules, whose aim is to manage critical situations (such as avoiding collisions, or leaving the area after ten seconds). The activation of these modules does not depend on the higher level activity.

In order to give an idea of how the whole system works, let us consider the case of the *AvoidObstacle* behavior. The related behavioral module contains some rules which become active when it is detected any obstacle on the path followed by the robot, and they produce actions which modify the current trajectory in order to avoid collisions. The problem that we must face with a full parallel behavior architecture is that the actions proposed by the *AvoidObstacle* module are composed with the output of other behavioral modules, thus producing unexpected or even undesirable results. One possible solution consists in disabling any other behavior module while the *AvoidObstacle* is active. This approach achieves the aim of avoiding collisions with other objects, but it has some drawbacks. For instance, when the *AvoidObstacle* module is the only active behavior module, it avoids the obstacles by taking a trajectory that is independent from the one that would have been produced by the other behavioral modules, thus degrading the global performances.

With the hierarchical organization of Mr.BRIAN, we place the *AvoidObstacle* module at the second level, so we can write rules of this kind:

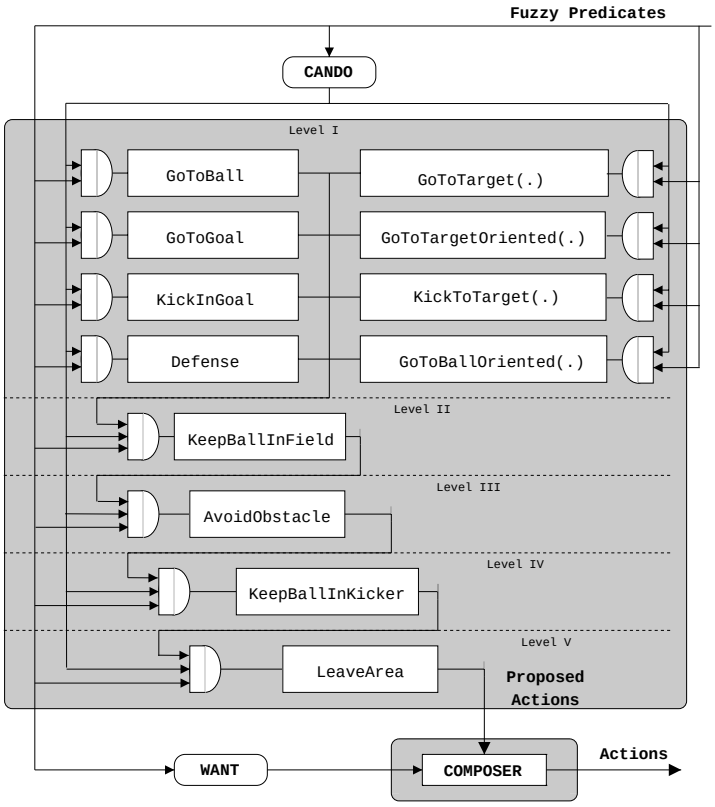

Figure 2: The behavior configuration used in RoboCup

(AND (ObstacleNordNear)
(SOMEBEHAVIOR.TanSpeed.*FORWARD)) ⇒ (DEL.TanSpeed.*FORWARD)
(TanSpeed STEADY)

that can be read as: "*if* I am facing any obstacle that is near *and* I would like to move forward *then* discard all the actions that propose a forward movement *and* propose to stay steady". Whenever the robot has no obstacle in its proximity, the *AvoidObstacle* module leaves all the proposed actions unchanged to the following level.

In Figure 3 we can see a sequence that shows Reseghé (one of our robot players) that goes to the ball while avoiding a couple of obstacles. Reseghé is an omnidirectional robot that can change its shape (i.e. it can pass from the large-and-short configuration to the narrow-and-long one) according to the current situation. The robot initially goes directly towards the ball until it comes close to the obstacle (Figure 3(a)), where the *AvoidObstacle* becomes active and deletes all the actions that would push the robot against the obstacle. The actions proposed by *GoToBall* that are not filtered by *AvoidObstacle* allow the robot to go to the left of the obstacle, where the *AvoidObstacle* decides to change shape in order to make easier the passage (Figure 3(b)). Then, the robot continues to run towards the ball (Figure 3(c)), but it must turn slightly to the right in order to avoid the second obstacle (Figure 3(d)). Finally the robots approaches the ball (Figure 3(d)) and changes its shape again (Figure 3(d)).

It is worth notice that this approach allows the *AvoidObstacle* to pass the obstacles to the right side for reaching the ball, even if it does not explicitly know that the goal of the robot is to catch the ball. This implies that we can reuse this behavior in any other application, without any change. This seems impossible with other behavior architectures.

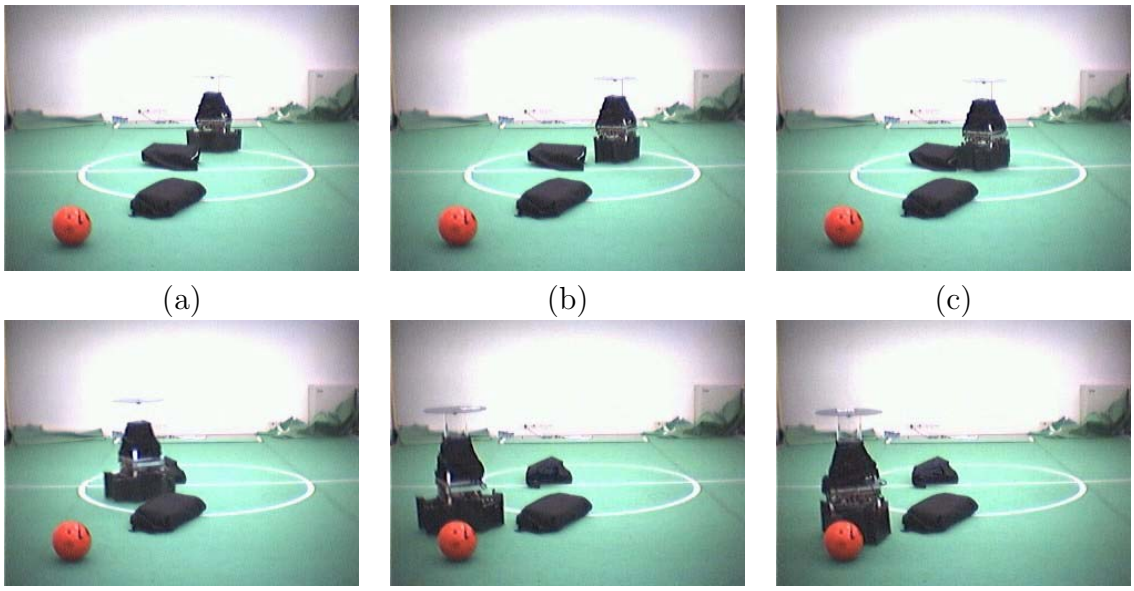

Figure 3: A slalom between obstacles

This design approach is in apparent contrast with the behavior independency principle, but it is the opposite. Consider the *KeepBall* behavior, taken again from the RoboCup domain. The goal of this behavior is to hold the ball inside the kicker while the robot is moving. When the robot takes the ball, it is of primary importance to maintain the ball possession, we must put the *KeepBall* module at the highest level, so that it can, if necessary, modify the proposed actions as much as it is needed for keeping the ball inside the kicker. In non-hierarchical architecture, the most effective way of implementing the *KeepBall* behavior is to embed it in each behavioral module, thus complicating the rules and really breaking the principle of modularity. In our architecture, the dependencies among modules are explicit by the hierarchical structure, and the interface is limited to the action proposed by the modules.

4 Conclusion

In this paper, we have proposed a new approach to the solution of the command fusion problem in behavior-based architectures. We propose to provide in input to behaviors, organized in a hierarchy of criticality, information about the actions proposed by less critical behaviors. Thus, behaviors can be designed taking into account, when possible, the actions proposed by others. This produces an overall behavior that exploits at best the knowledge contained in all the behaviors. Moreover, the designer can follow the "independent design" principle of the behavior-based approach, by designing behaviors without taking care of the contents and even the presence of others: only potentially disturbing actions are considered when designing a behavior. This approach solves some of the problems left unsolved by most of other behavior-based approaches.

In a next future, we plan to make the hierarchy changing according to context, by shifting the behaviors from one level to another. This is possible, since our behaviors are independent from any link with others. It has also some relevance in some dynamical environments, where the criticality of a behavior changes with the characteristics of the

environment. For instance, in Robocup, sometimes referees are more strict in judging contact with other robots as charging fouls: by moving the AvoidObstacle behavior down in the hierarchy, we obtain a robot less respectful of the opponent, but more effective in controlling the ball and dribbling. This dynamical change of level can be also related to an automatic adaptation mechanism that may reason on the environment and organize behaviors accordingly. We are already using a similar adaptation mechanism to adapt the team strategy to opponent's characteristics [4].

References

[1] R. C. Arkin. Towards the unification of navigational planning and reactive control. In *Proceedings of the AAAI Spring Symposium on Robot Navigation*, pages 1–5, 1989.

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

[3] A. Bonarini, G. Invernizzi, T.H. Labella, and M. Matteucci. An architecture to coordinate fuzzy behaviors to control an autonomous robot. *Fuzzy Sets and Systems*, 134(1):101–115, 2002.

[4] A. Bonarini and M. Matteucci. Learning context motivation in coordinated behaviors. In *Proceedings of IAS-6*, pages 519–526, Amsterdam, NL, 2000. IOS Press.

[5] A. Bonarini, M. Matteucci, and M. Restelli. Filling the gap among coordination, planning, and reaction using a fuzzy cognitive model. In *RoboCup 2003 - Robot Soccer World Cup VII*, Berlin, D, 2003. Springer Verlag.

[6] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robot Automation*, 2(1):14–23, 1986.

[7] L. Correia and A. Steiger-Garc ao. A useful autonomous vehicle with a hierarchical behavior control. In Merelo J.J., Mòran F., Moreno A., and Chàcon P., editors, *Advances in artificial life, 3rd European Conference on Artificial Life*, pages 625–639. Springer, 1995.

[8] P. Pirjanian. Behavior coordination mechanisms - state-of-tha-art. Technical Report IRIS-99-375, Institute of Robotics and Intellient Systems, School of Engineering, University of Southern California, October 1999.

[9] 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.

[10] A. Saffiotti and Z. Wasik. Using hierarchical fuzzy behaviors in the robocup domain. *C. Zhou, D. Maravall and D. Ruan (Eds) Autonomous Robotic Systems. Springer, Berlin*, pages 235–262, 2003.

[11] E. Tunstel, M.A.A. de Oliveira, and S. Berman. Fuzzy behavior hierarchies for multi-robot control. *International Journal of Intelligent Systems*, 17:449–470, 2002.