Concepts and Fuzzy Models for Behavior-Based Robotics

Andrea Bonarini, Matteo Matteucci, and Marcello Restelli

Politecnico di Milano Artificial Intelligence and Robotics Lab,
Department of Electronics and Information,
Politecnico di Milano, Milan, Italy
Phone: +39 02 2399 3525 – Fax: +39 02 2399 3411
{bonarini, matteucc, restelli}@elet.polimi.it

Abstract. In this paper, we propose a modeling paradigm that uses fuzzy sets to represent concepts on which control modules of a behavior-based autonomous robot operate. The primitives defined in the modeling paradigm are expressive enough to represent the knowledge needed by planning, coordination, and reactive control of a multi-robot control system. At the same time, it provides a well-founded tool to represent in a compact way the data interpretations, needed to reason effectively about what is happening in the world and what is desired to happen. This modeling paradigm makes the design of behavior, planning, and coordination modules easy, since its primitives are simple and expressive.

1 Introduction

Since some years, the most common architectures of autonomous robots integrate the planning activity, which provides goals for the robot, with behavior-based reactivity, which implements simple and fast control modules. In designing this kind of hybrid architectures, most of the issues arise from the connection between the conceptual and physical level representations used respectively in the deliberative and reactive components of the system. Although this practice is now common, only few efforts have been done to formalize, unify, and optimize the knowledge representation model in order to seamless integrate the components in the hybrid architecture.

In this paper, we present our approach to knowledge modelling for autonomous robots, aimed at providing a common framework to represent all the knowledge needed by the modules that participate in control and coordination. We define the conceptual aspects needed to represent this type of knowledge and we introduce fuzzy sets as a tool to support this representation. This fuzzy conceptual representation is used by all the modules of our control architecture: *MAP* (*Map Anchors Percepts*) [4] that integrates data from sensors and other data sources building an internal representation of the world, *BRIAN* (*Brian Reacts by Inferential ActioNs*) [3] that manages the behaviors and implements all the reactive functionalities of our system, and *SCARE* (*Scare Coordinates Agents in Robotic Environments*) [5] that coordinates the agent's behaviors and plans its activity. In fact, a uniform knowledge representation makes it possible a coordinated design of the modules, and an efficient exchange of information.

V. Di Gesú, F. Masulli, and A. Petrosino (Eds.): WILF 2003, LNAI 2955, pp. 72–79, 2006.

2 Concepts and the Fuzzy Model

In a robotic environment, agents have to interact with several *physical objects* and this interaction is typically implemented as a perception-action loop. Robots are equipped with sensors perceiving physical characteristics of the environment and they use these *percepts* to build an internal representation of the environment. Once this internal representation is formed, it is possible to use it for deliberative or reactive processing which produces actions to be executed in the environment. In this paper, we do not go into the theoretical details related to the problem of matching percepts with the corresponding semantic meaning of the physical objects. Instead, we focus on the knowledge representation we use to face the problem of creating, and maintaining in time, the connection between symbol-level and signal-level representations of the same physical object [4].

We propose to use a two stages process for creating the agent internal representation of the environment: first of all percepts are used to instantiate a real-valued conceptual model of the environment and then this conceptual model is interpreted in terms of fuzzy predicates to be used in coordination and control.

Percepts are processed by sensing modules (i.e., smart sensors) to produce high level features. Features referring to a specific physical object are collected around the same internal representation, referred to as its *perceptual image* and it can be seen as the instance of a *concept*. In a formal way, a concept C is described by a set of properties defined as tuples in the form $p \triangleq < label, \mathbb{D}, \rho >$, where $label$ denotes the property name, $\mathbb{D}$ is the set of all the possible values for that property given a specific representation code (e.g. for the colors we can use the set $\{red, green, blue, \ldots\}$, or the RGB space $\mathbb{N}^3_{[0,255]}$, or a fuzzy classification of this) and ρ represents a restriction of the domain $\mathbb{D}$ for the property in the specific concept.

According to this concept-based knowledge representation, a property can be either directly perceived, and thus related to a set of high level features coming from sensors, or it can be derived from other properties through inference or computation. This approach allows properties specific to the concept to provide additional information about the perceptual image, or to infer unsensed characteristics.

Depending on the concept and on the specific application domain, a property can be classified as *substantial* or *accidental*. Substantial properties characterize the immutable part of a concept: for a given object, their values do not change over time, and they can be used for object recognition since they explain the essence of the object they represent. Accidental properties do not characterize a concept: their values are specific to each conceptual instance, and they can vary over time. They cannot be used for object recognition, but are the basis of instance formation, tracking and model validation. During robot activity, data coming from sensors are matched against concepts in the conceptual model, and, when enough evidence is collected, a concept instance is generated, which inherits by default property values eventually not detected by sensors [4].

Using concepts it is possible to describe both domain specific and general knowledge used by an agent during its activity. To explain how this knowledge is used, we introduce the notion of model $\mathcal{M}$: given D as the set of the known domains, a model $\mathcal{M}_d$ is the set of all the concepts known by the agent referring to the specific domain $d \in D$, linked by relationships – structural (e.g., generalization, and specialization) and domain specific (e.g., colors and landmark in structured environments).

The definition of concepts at different levels of abstraction is important to support the classification of percepts, and the instantiation of concepts. Concepts are organized in *ontologies*, which may be partially defined independently from the specific application, at least up to a certain abstraction level. For instance, it is possible to give general properties of *movable objects*, as a concept specializing the more general concept *objects*, and in turn specialize it in *mobile robots* and *human beings*. Such general concepts may also participate in general inferential processes, which allow, for instance, to infer that people and mobile robots usually stay on the ground, information useful to compute distances from images. When facing a specific application, it is then possible to complement the general ontology with application-specific information; for instance, we may already know that balls are spherical, but in a Robocup [2] robot soccer application we also know that the ball is red and has a given dimension. In case of uncertainty, it is often more reliable to instantiate a more general perceptual image. For instance, again in a Robocup application, robots belonging to different teams wear different markers, which may be detected with some uncertainty; therefore, a set of features may be aggregated more reliably as an instance of *robot*, than as an instance of *opponent robot*.

From the design point of view, the presence of a reference model makes it possible a modular design, with people having different competence interacting on the same knowledge. People working on sensors would know that they should produce certain information in a given format, and people working on control could rely on that. Once the conceptual model $\mathcal{M}$ relative to the agent's established knowledge has been instantiated, we have an internal representation of the environment on which it is possible to evaluate logical predicates, apply inference, or execute behavior control modules. We call this internal representation *domain* and we will denote it with $\mathcal{D}$.

Fuzzy predicates may represent concept instances related to aspects of the world, goals, and information coming from other agents. They are represented by a *label* λ, its *truth value* μ_λ, computed by fuzzy evaluation of the concept instance properties, and a *reliability value* ξ_λ to take into account the quality of the instance. For instance, we may have a predicate represented as $< ObstacleInFront, 0.8, 0.9 >$, which can be interpreted as: "It is quite true ($\mu_\lambda = 0.8$, coming from the fuzzyfication of real-valued properties) that there is an obstacle in front of the robot, and this statement has a reliability quite high ($\xi_\lambda = 0.9$, due to the reliability of the sensing conditions)"

We consider ground and complex fuzzy predicates. *Ground fuzzy predicates* range on concept properties directly available to the agent through $\mathcal{D}$, and have a truth value corresponding to the degree of membership of instance properties to labeled fuzzy sets. The reliability of sensorial data is provided by the anchoring process basing on percept analysis, and goal reliability is stated by the planner. A *complex fuzzy predicate* is a composition of fuzzy predicates obtained by fuzzy logic operators. Complex fuzzy predicates organize 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 *BallOwner* predicate, defined by the conjunction of the ground predicates *BallNorth* and *BallInKick*, respectively deriving from the fuzzyfication of the direction and distance properties of the ball concept instance in $\mathcal{D}$.

Some of the most important properties that can be obtained by basing the robot control architecture on the model using the knowledge model are below summarized.

- *noise filtering*: using a conceptual model of the environment it is possible to eliminate out–layers in percepts and filter in a proper way noisy data coming from sensors; this produces more reliable information for the other modules effectively controlling the agent
- *sensor fusion*: percepts coming from different sensors, and referring to the same objects, can be fused enhancing fault tolerance and enabling on–line diagnosis
- *virtual sensing*: a model of the environment can be used to infer new features, not perceived by physical sensors
- *time consistency*: the instances in the conceptual model represent a state of the environment; it is possible to maintain and monitor its consistency in real–time; this activity can be used to learn and check models
- *abstraction*: the use of fuzzy predicates instead of raw data, or features, in the behavior definition gives more abstraction in designing robot behaviors, and robustness to noisy data; it also facilitates design, since gives the designer the possibility to reason in symbolic terms. Moreover, exchanging this information is more effective for agents of a Multi-Agent System (MAS) sharing the same semantics for symbols.

3 SCARE, the Coordination System

Cooperation holds a very important role in multi-agent system applications. To face the typical issues of these applications, we have implemented *SCARE* [5] a general architecture for coordination in multi-robot domains. SCARE is able to deal with:

- *heterogeneity*: when a MAS is made up of agents with different skills, our architecture exploits these differences in order to improve the overall performance
- *communication*: coordination policy may change according to the amount of information that can be exchanged among agents and according to network connectivity
- *adaptation*: in order to grant the autonomy of the system, the coordination mechanism is able to adapt its parameters in reaction to environment changes

Using SCARE, the MAS application developer has to identify the macro-activities that the agents can carry out. To cope with uncertainty of the perception and approximate definitions, we adopt the *fuzzy predicates* approach introduced in Section 1. In this way, the states of the model belong to a situation with a certain degree. By introducing a threshold $\overline{t}$ it is possible to define a situation with a fuzzy predicate: $\sigma = \{m \in \mathcal{M} | \mu(m) > \overline{t}\}\ \ \overline{t} \in [0, 1]$. Fuzzy predicates give the possibility to obtain a measure of the matching between a predefined situation σ and the current state of the model. In the job assignment process, in order to establish how much each activity is suited for the agent, we use several parameters implemented by fuzzy predicates, which operate on the domain $\mathcal{D}$:

- *cando*: define when the activity can take part in the assignment process;
- *attitude*: define how much the skills of the agent are useful for the activity;
- *chance*: define the situation where the agent has good chances to succeed;
- *utility*: define the situation where the activity is useful for the agent team;

- *success*: define the goal achievement situation for the activity;
- *failure*: define the situation where the activity should be stopped because of unrecoverable failure.

An activity terminates when the success or failure conditions are verified. If an agent is idle, a job assignement process starts. For each activity, the cando predicates are evaluated in order to reject those activities that cannot take place. For each remaining activity, utility and chance predicates, and the agent's attitude are considered, thus obtaining indicators to take the decision. Through the application of some multi-objective technique (e.g., weighted sums, goal programming, normal-boundary intersection, multilevel programming, or others), each agent gets an ordered list of activities (*agenda*). Once all the agendas are produced they must be compared in order to achieve a coordinated assignement of jobs (for details see [5]).

4 BRIAN: The Behavior Management System

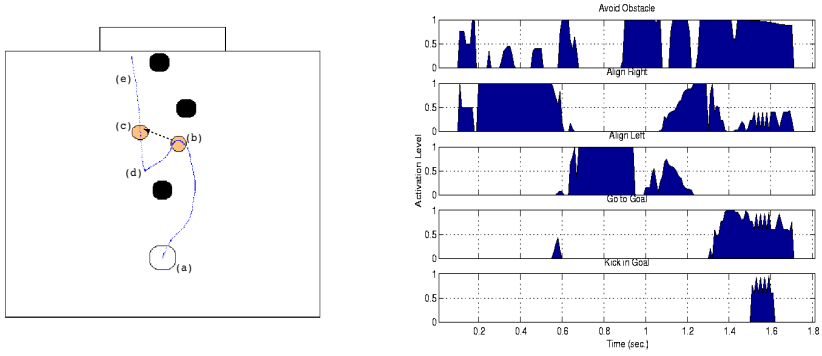
Especially in dynamic environments, the main approach to robot control design is the so called behavior-based architecture [1]. In such approach, the robot controller is obtained by the implicit cooperative activity of behavioral modules. Each module operates on a small subset of the input space implementing a relatively simple mapping from sensorial input to actions; the global behavior of the robot arises from the interaction among all these modules. One of the major problems in behavior-based robotics is the design of this interaction, usually pre–defined in terms of inhibitory relationships or vectorial composition of the module output.

In our behavior management system *BRIAN*, integration and coordination among behavior modules is achieved using two sets of fuzzy predicates associated to each of them: *CANDO* and *WANT* conditions. In BRIAN we face the issue of controlling the interaction among modules by decoupling them with context conditions described in terms of fuzzy predicates evaluated over internal state and environmental situation present in $\mathcal{D}$, goals generated by *SCARE*, and communications with other agents.

CANDO conditions are used to decide whether a behavior module is appropriate to the specific situation: if they are not verified, the behavior module activation does not make sense. The designer has to put in this set all the fuzzy predicates 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, and it should be oriented towards the goal. This set of conditions has a twofold result: decoupling behavior design and increasing the computational efficiency of the behavior management system.

WANT conditions represent the motivation for an agent to execute a behavior. They may be related either to the environmental context (e.g., *BallInFront*, "the ball is in front of me"), or from strategic goals (e.g., *CollectDocuments*, "I have to collect the documents to be delivered"). Composition of the actions proposed by behaviors modules active at the same time is implemented by the WANT conditions, which represent the opportunity of executing them in the specific context.

The use of these two different sets of conditions allows the designer to design a dynamic network of behavior modules defined by means of context predicates. This is



(a) (b)

Fig. 1. The trace of robot trajectory during the test and the activation level of the behaviors

a sensible improvement with respect to usual behavior-based architectures; 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 only a restricted set of behavior modules (i.e., those enabled by the CANDO conditions), and it has to select/merge the behaviors consistent with its present motivations.

In BRIAN, each behavioral module receives data in input and provides output commands to be issued to the environment. This is obtained through fuzzy logic controllers where a set of fuzzy rules match a description of the situation given in terms of fuzzy predicates, and produces actions for the actuators by composing the output proposed by each rule by a T-conorm.

5 Experimental Results

In this section we present in details the results obtained in a particular experiment where the robot executes a standard Robocup challenge used in this case to test the composition of behavioral modules.

In figure 1(a) you can see the experimental setting of the test. The black objects are static obstacles the grey one is the ball to catch and kick in the goal. The track in the plot is taken from real data: the trajectory is projected on the field using the odometry of the robot. The robot starts in position (a), after 0.5 sec the robot, incidentally touching the ball (position (b)), moves it to the position (c). So it has to dynamically change its behavior to face this unforeseen situation. The trajectory executed by the robot is obtained by selecting and blending different actions proposed by different behavioral units and reacting to changes in the environment (e.g., the variation of ball position as shown in figure $1(b)$). In this experiment, we use only five simplified basic behaviors to show the effectiveness of our approach (see Table 1).

The predicates appearing in the fuzzy conditions are computed by evaluating fuzzy sets on data in the domain $\mathcal{D}$, and composing them. In figure 2, you may see the definition of some of the fuzzy sets involved in the definition of the above mentioned

Table 1. CANDO and WANT conditions for each behavior

Behavior	CANDO	WANT
Avoid Obstacle	`(ObstaclePresent)`	
Align Right	`(AND (AND (BallSeeing) (NOT (RigthAligned))) (NOT (AND (BallOwner) (Aligned))))`	`(AND (OR (AlonePlayer)(ForwardRole)) (AND (NOT (ObstacleAvoiding)) (NOT (GoalNear))))`
Align Left	`(AND (AND (BallSeeing) (NOT (LeftAligned))) (NOT (AND (BallOwner) (Aligned))))`	`(AND (OR (AlonePlayer)(ForwardRole)) (AND (NOT (ObstacleAvoiding)) (NOT (GoalNear))))`
Go to Goal	`(AND (BallOwner)(Aligned))`	`(AND (NOT (AbstacleAvoiding)) (NOT (GoalNear)))`
Kick in Goal	`(AND (BallOwner)(Aligned))`	`(GoalNear)`

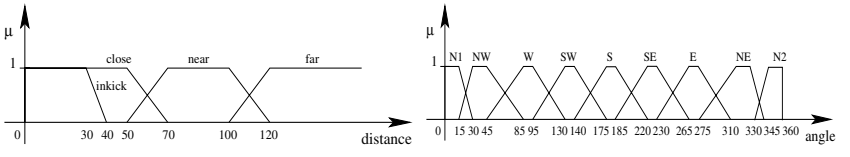


Fig. 2. The definition of some of the fuzzy sets used to compute the predicates involved in CANDO and WANT conditions of the behavior modules developed for Robocup

predicates. In particular, we have reported the frame of cognition the distance to the ball, and the direction of the ball.

Complex predicates are computed by composing basic ones. For instance, the `BallOwner` predicate is computed as `(AND BallNord BallInKick)`, where `BallNord` comes from `(OR N1 N2)` computed from the second frame of cognition reported in figure 2, and `BallInKick` comes from the value of `InKick` from the distance frame of cognition. Analogous computations bring to the evaluation of all the needed predicates.

During the experiment we are presenting, we have logged the activation level of CANDO and WANT for each behavior module (figure 3(a) and (b)) and the final behavior activation level coming from the combination of CANDO and WANT (figure 1(b)). From figure 3(b) you may notice that, during this experiment, the WANT conditions for *Align Left*, *AlignRight* and *GoToGoal* have the same activation level, since the robot is playing alone so the predicate `(AlonePlayer)` is always verified and the other conditions are the same for all the behaviors.

As you may notice from the definitions of the WANT conditions of *AlignLeft* and *AlignRight* in Table 1, it is possible to include in them also context description predicates like the actual role of the teammate (i.e., `ForwardRole`) or the explicit reference to other behaviors (i.e., `ObstacleAvoiding`). In the example, the value of `ForwardRole` has been set by the user to be always `TRUE`; in a real sistuation SCARE sets this value according to the evaluation of current situation and the planning strategy. This is just an example of how our formalism can be used both for modeling the context and controlling the behaviors blending.

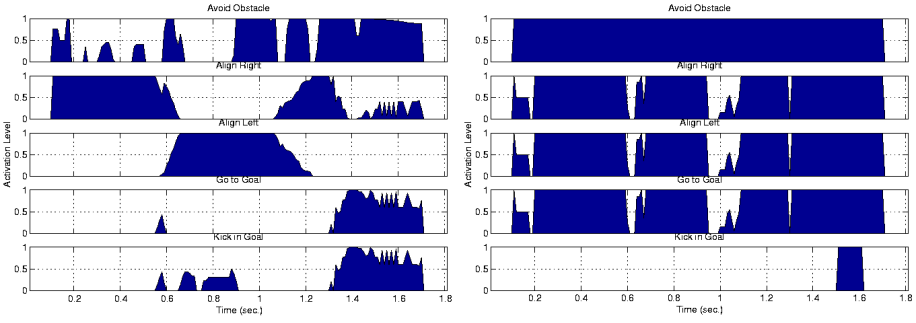


Fig. 3. The activation level of CANDO (a) and WANT (b) conditions during the test

6 Conclusion

In this paper we have presented the fuzzy cognitive model we use to integrate in a uniform framework the deliberative and reactive components of multi-agent systems. The cognitive model we propose, integrates coordination, planning and reactive behaviors providing a common cognitive substratum for a team of robots where behavioral modules are used as high-level macro-actions that compose structured plans defined by a flexible multi-agent coordination system.

All the elements in the knowledge processing level are based on simple fuzzy predicates that can be easily managed, designed and adapted. Doing it this way, the control model can be easily designed to be tuned and adapted on-line so that the team strategies and the role of robots in the control schemata can be automatically modified to face different opponent teams, and changes in robot performances [5].

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, G. Invernizzi, T. Labella, and M. Matteucci. An architecture to co-ordinate fuzzy behaviors to control an autonomous robot. *Fuzzy Sets and Systems*, 134(1):101–115, 2002.
4. A. Bonarini, M. Matteucci, and M. Restelli. Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem? In *Proceedings of the AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, page In press, Menlo Park, CA, 2001. AAAI Press.
5. A. Bonarini and M. Restelli. An architecture to implement agents co-operating in dynamic environments. In *Proc. of AAMAS 2002 - Autonomous Agents and Multi-Agent Systems*, pages 1143–1144, New York, NY, 2002. ACM Press.