A Reactive Architecture for RoboCup Competition

E. Pagello[1,2], F. Montesello[1], A. D'Angelo[3], C. Ferrari[1]

[1] Dept. of Electronics and Informatics, Padua University, Italy
[2] Inst. LADSEB of CNR, Padua, Italy
[3] Dept. of Mathematics and Informatics, Udine University, Italy

Abstract. We illustrate PaSo-Team (The University of Padua Simulated Robot Soccer Team), a Multi-Agent System able to play soccer game for participating to the Simulator League of RoboCup competition. PaSo-Team looks like a partially reactive system built upon a number of specialized behaviors, just designed for a soccer play game and generating actions accordingly with environmental changes. A general description of the architecture and a guideline of main ideas is presented in the paper, whereas a more detailed description of actual implementation is given in the appendix.

1 Introduction

We have experienced that a soccer game should successfully deal with the reactive phase before any attempt to explore the reasoning and planning phase of the game. This approach stems from the observation that at each time of the game, *scoring a goal* can be easily recognized by every player to be an individual target, besides a global one. This allows the single agent to try, whenever possible, to score the goal directly. This is not the case of a situation such as a robot engaged to reach a fixed location in a well structured environment, like a floor inside a building with many offices. Before starting any useful action, such a robot should acquaint with the world building a map it shall use later to find out how to fulfil this task. Thus, it needs a planning phase because it has no way to know immediately what action to perform towards the achievement of the goal. In the simulated soccer game, instead, at each time of the game every player knows how to do to realize its task, by using only perceptive information. This means that it is possible to consider every player to be able to bring to completion his own task. In this perspective we have faced the design of a MAS starting with a reactive approach.

In this paper we illustrate how we solve the problem of coordination among agents using an implicit communication approach whose motivation stems from the simplified assumption that a number of low level reasoning capabilities can be endowed into the system..

2 Arbitration and Implicit Coordination

As it has been pointed out in literature [1] a sound arbitration mechanism is the base for an appropriate performance of a behavior-based autonomous sys-

tem. In our case a further difficulty arises, due to the simultaneous presence of several playing agents in the same environment. Starting from the pioneristic subsumption architecture originally devised by Brooks [3], a number of innovative behavior-based systems have been proposed in literature (Connell [4], Maes [7], Anderson [1], Kaelbling [6].

Their proposals are dominated by the concept of arbitration which results in an either spatial or temporal ordering of behaviors. The former causes the concurrent activation of a set of primitive reflexive behaviors, also referred to as static arbitration, the latter brings about a sequential activation of different sets of primitive reflexive behaviors, also referred to as dynamic arbitration.

However, because the inclusion of temporal ordering appears too problematic when it is devised within a general multiagent framework, we have implemented a static arbitration as a special purpose behavioral module where pre-processed sensor data are always channeled to discriminate a candidate skill to be enabled as a response to typical perceived patterns. Every time sensor data are directly channeled between the perception block and the selected behavior, this behavior is activated whereas the remaining ones are inhibited.

The resulting architecture, shown in fig. 1, resembles partly the proposal of Anderson and Donath [1] and partly that of D'Angelo [5] in what the collection of boolean values (flags), updated using information supplied from sensor data pre-processing, defines a coarse- grained global state of an agent which controls behavior switches as a rough inhibitor/activation mechanism.

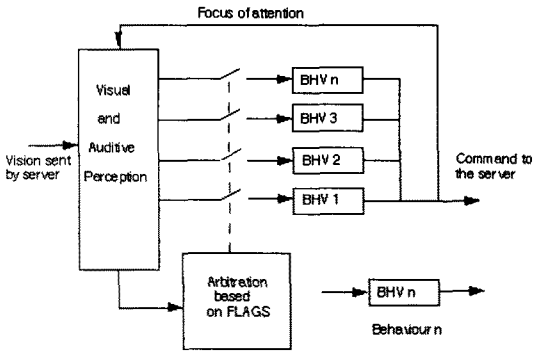

Fig. 1. agent arbitration

At each time of the game, every team player will be enabled with a behavior which depends on both its current position and orientation in the soccer game field and the pattern of the objects the agent is aware of. So, though every agent has the same cloned structure, it does not need to be activated with the same behavior. This means that is the world that makes differences among the agents.

Now how can we get any *coordination* among agents equipped with this

sketched architecture? A MAS fully distributed architecture that uses a behavior-based approach is proposed by Parker [9] that consider only the cooperative side of coordination among heterogeneous mobile robots, with attention to fault tolerance.

Another approach is proposed by Rude [10] with the IRoN architecture. He considers two kind of cooperation among robots, namely via implicit or explicit communication. With implicit and explicit communication we intend, as in Rude a passing of information respectively non-intentional and intentional. The first is realized "looking" at the external behavior of the other agents, without those agents "planned" to transmit whatever information. The second is realized sending volunteerly explicit coordination massage to the other agents. In our approach we intend to fully exploit only the *implicit communication* to perform a higher-level quality of cooperation and competition too.

3 The Basic Behaviors

The behaviors used in building the body structure of the agents have been developed to capture the abilities of a real soccer player. We have mainly tried to reproduce low-level skills.

In a soccer game individual skills are the base that every team needs before trying any kind of team strategy. Furthermore, we have noticed that the subgoal of scoring a goal is performed, at last, by a single player and this means that though it is wrong to consider a soccer as a single-player game, the acquisition of individual skills becomes a primary task.

The system implemented by Veloso and Stone [11] allows the players to learn low-level skills, as shooting to the goal, or intercept the moving ball, using neural networks. On the contrary, we have chosen an analytical approach, using the motion laws featured by the server to realize the simulation. The behaviors are described in appendix.

4 The Coordination Model

To extract significant information useful for coordination and representing a sort of *global* state of the agent or of the system, we use, immediately after every received visual or auditive message, a function named *estimate_state*, that builds for each player the data structure representing the world as the player saw it last time. Every player needs an individual execution of this function, because every player has his own view of the world. This data structure contains the last absolute seen positions, speed and orientation information of each mobile object in the field and other useful information. Using this data structure a player is able to extract, via dedicated functions, the flags used in the arbitration, like *kickable*, *ball_stolen*, *near_ball*, *passage*. Those represent individual states for a single agent. Another flag is significant to understand the arbitration: the *attack/defence flag*. This flag, that is one of the descriptors of the *global* state of the system, beside

that of the agents, is used to identify the attack or defence states of the team *(attack/defence playmode)*. The flag is set when a team-mate become owner of the ball. Subsequently the same player sends an auditive command to all the players presents in the hearing area. This command, sent to the server, causes an auditive perception to all the players standing up within a 50 meters radius circle, and consequently sets the attack/defence flag.

The basic coordination mechanisms induced by the arbitration involve usually two behaviors at a time. In the attack playmode, such behaviors characterize two players belonging to the same team, that is the ball-holder and a potential receiver of the ball. We want to realize the simultaneous activation of the behavior *playball* for the player with the ball and the behavior *smarcamento* for the next ball-holder candidate. The coordination arises through the simultaneous activation of the pair {bhv_1, bhv_2} for any pair of players, where, during attacking, bhv_1 = *playball*, bhv_2 = *smarcamento*, whereas, during defending, bhv_1 = *bhv_X*, bhv_2 = *interdict*. The behavior *bhv_X* is not a real implemented behavior, but it is referred to the apparent behavior of the opponent. Actually, the opponent estimated behavior is *chasing the ball*, so our player must compete against it.

To realize this kind of interaction we have built a rigid arbitration which choose a candidate behavior looking at the *global flags* representing particular states of the whole team of agents, and at some *local flags* related to states of a single agent. In this way the emerging cooperative behavior appearing during the game may be considered as an *eusocial behavior*, a collective behavior due to the interaction of "genetically" determined individual behavior, as discussed in McFarland [8]. The proposed arbitration is an hint for the emerging of a social behavior, like an ant colony that seems to be a relatively smart being even if formed by a finite number of pure instinctive individuals. The emergence of some sort of intelligent behavior like triangulations or non explicit pass arise mainly from the interaction between the single players and the environment, namely in our case between our players and the opponents, exploiting advantageously their dynamics, as shown in fig. 2. In (1) an enemy defender chase the ball (hypothized reasonable behavior assumed by the enemy player nearest to the ball) owned by a friend player activated on the *playball* behavior. In this way the defender creates a free area in its rear. So in (2) a forward, activated on the *smarcamento* behavior, find the new hole (closed dotted line) and go to take position. With (3) the coordinated action is closed. The ball's owner see the forward in a good position and make the pass. Without communication.

There are other two kind of coordination realized, this time only among our players. The first concerns the coordination among our player in defence play mode. When the opponents are playing the ball, our players apply a negotiation mechanism to select the teammate that must make pressing, chasing for the ball. This lets the other teammates free to assume another useful defensive behavior like *interdict*.

The second one concerns the coordination of the whole team, during the game. It is realized changing dynamically the default_pos of every player in the

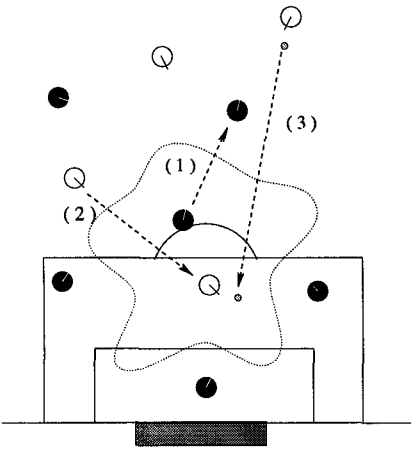

Fig. 2. observed *emergent* pass

team, accordingly with the ball position. This drag all the players, according to their roles, to follow the ball in case of attack or to back gradually to the defence position in case of defence.

5 The PaSo-Team

A full description of the implementation of PaSo-Team Clients (The University of Padua Simulated Robot Soccer Team) is given in [2]. For a brief description see the appendix.

6 Conclusions

We started by assuming that the main characteristic of soccer game is highly reactive, and from the fact that in the robot soccer competition all clients have a clear understanding of their task. We solved the problem of coordination among agents using an implicit communication approach, without using any form of reasoning about agent's intentions. We used this approach for designing PaSo-Team, our multi-agent system for RoboCup, relying on the following criteria:

- The coordination among agents is realized through a cooperative or competitive interactions respectively with team-mates and opponents.
- Flags setting gives flexibility to PaSo-Team performance, allowing to generate different soccer team able to play games with different ability.
- Arbitration is done separately over each agent, but cooperative coordination is obtained between pair of agents by ball exchanges, among several agents by pressing negotiating, and over all team by dynamically changing default positions.

- The team architecture really exploits the interaction dynamics among the agents and the environment, and if possible exploits the emergent collective behavior, without using explicit communication to realize coordination.

Acknowledgements

This research could not be done without the enthusiastic participation of the students of Electronics and Computer Eng. Undergr. Div. of Padua University Eng. School. Financial support has been provided by both CNR, under the Special Research Project on "Real-Time Computing for Real- World" and Murst, under the 60% Grants. A particular thank is due to the Industrial Firm Calearo S.R.L., a car aerials and cables manufacturer, located in Isola, Vicenza(Italy), that have provided coverage of all expenses for participating at IJCAI Conference. We like to thank also Padua Branch (Italy) of Sun Microsystems, that provided us freely a SUN ULTRA1 for developing PaSo-Team.

References

1. T.L. Anderson and M. Donath. Animal behaviour as a paradigm for developing robot autonomy. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 145–168. The MIT Press, Cambridge (MA), 1990.
2. F. Bidinotto, A. Bissacco, M. Dal Santo, W. Frasson, S. Griggio, A. F. Grisotto, S. Marzolla, F. Montesello, and E. Pagello. Implementing a soccer client team for robocup '97 competition. Technical report, LADSEB-CNR, Padua (I), 1997.
3. R. Brooks. A layered intelligent control system for a mobile robot. *IEEE J. on Rob. and Aut.*, RA-2:14–23, Apr. 1986.
4. J. H. Connell. *Minimalist Mobile Robotics.* Number 5 in Perspective in Artificial Intelligence. Academic Press, 1990.
5. A. D'Angelo. Using a chemical metaphor to implement autonomous systems. In M. Gori and G. Soda, editors, *Topics in Artificial Intelligence*, volume 992 of *Lecture Notes in A.I.*, pages 315–322. Springer-Verlag, Florence (I), 1995.
6. L. P. Kaelbling and S. J. Rosenschein. Action and planning in embedded agents. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 35–48. The MIT Press, Cambridge (MA), 1990.
7. Pattie Maes. Situated agents can have goals. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 49–70. The MIT Press, Cambridge (MA), 1990.
8. D. McFarland. Towards robot cooperation. In *From Animals to Animats 4, Int. Conf. on Simulation of Adaptive Behavior (SAB-94)*, Brighton, 1994.
9. L.E. Parker. Alliance: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proc. of IROS'97*, pages 776–783, 1994.
10. M. Rude, T. Rupp, K. Matsumoto, S. Sutedjo, and S. Yuta. Iron: An inter robot network and three examples on multiple mobile robots' motion coordiantion. In *Proc. of IROS'97*, pages 1437–1444, Grenoble, Sept. 1997.
11. P. Stone and M. Veloso. A layered approach to learning client behaviours in the robocup soccer server. In *http:// www.cs.cmu.edu/ afs/ cs/ user/ mmv/ www/ papers/ AAI96.ps.gz*, 1997.

Appendix

Co-authored by *F. Bidinotto, A. Bissacco, M. DalSanto, W.Frasson, S. Griggio, A.F. Grisotto, S. Marzolla, F. Montesello, E. Pagello* - Dept. of Electronics and Informatics, The University of Padua, Italy

Client Structure

Our software client built for RoboCup competition is composed of various elements, each working at different steps. An important aspect to care of is the reception and transmission of commands from and to the server, that takes place using a socket. The communication via socket requires a continuous control of the state of the socket itself, because of the unpredictability of the exact instant of the visual perception sent by the server. Each client controls the socket looking at a SIG-IO, given by C++ library, that is activated when messages are present in the receiving queue. A signal handler pops the message from the queue and sends it to the parser which extracts the suitable information. All communication are realized using the UDP/IP protocol.

Every behavior, activated by a client, generates commands for the server, that are stored in another queue, the command list, using the same protocol. The timing of reading this list is provided every 100 ms by a timer synchronised with the SIG-IO controlling the socket. In such a way, commands are sent to the server without loosing a single command.

Estimators

When the information are received from the parser, which stores the relative visual information for each player, another function, named *estimate_state* extracts from the relative data the absolute ones. These data allow to reason about the absolute *position*, *orientation* and *speed* of the players and every other object in the field.

The estimate of the positions and orientation, using the relative information sent by the server, is done with a geometrical approach, looking for every fixed object and making triangulation to recover the absolute ones. Extracting this information is not simple, because sometimes, when the player is in the neighbourhood of field border, the only visible objects are the side lines, which provide too few information to allow to retrieve a reasonable estimate.

Therefore, tree different situations are evaluated, relatively to the available relative visual information:

- The first case happens when a player watches two fixed objects. This allows a good estimate of the absolute position followed by the estimate of the absolute orientation
- The second case happens when a player watches only one fixed object and a border line. Then, we estimate first the absolute orientation and later the absolute position. The estimate is worse than the previous case.

– The third case, the worst one, happens when no fixed objects are visible, or one single fixed object only is available, but no border lines. Then, it is impossible to retrieve whatever kind of estimate. Thus, we use the memory of the past sent commands to estimate the current position from the available last one.

For acting in and reacting to the environment we used both absolute and relative information. For chasing the ball, turning with ball, we used relative information. We used the relative approach for this skills because of the better accuracy of the estimated values. With this approach we can evaluate the trajectory of the moving objects (usually the ball) predicting the future positions of the ball itself and trying to intercept or control it.

Behaviors

The following ones are the behaviors implemented to provide the interaction with the environment. The only behavior that does not generate commands for the server is the *arbitrate* behavior. The remaining behaviors are all sending commands.

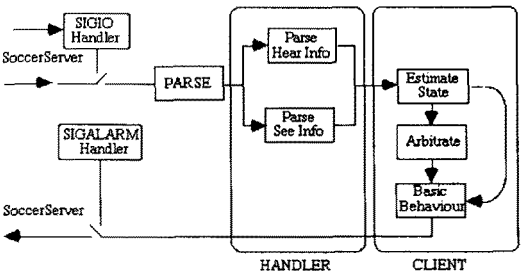

Fig. 3. agent architecture

arbitrate() It is always active. It analyses the flag structure and activates the right behavior. It is implemented in a rigid manner, actually using sequences of **if** statements.

playball() It provides a player with the commands able to realize the correct actions when it owns the ball.

This means:

- 1. taking the ball in a *free from enemies* direction
- 2. deciding if passing or not the ball
- 3. testing dangerous situations. If running with the ball (or passing it) becomes too dangerous, then the client throws the ball in a *free from enemies* direction.

chase() It forces the player to chase the ball.

This may happen in two ways, named stamina-preserving and stamina-consuming. The former is invoked, when the ball is far whereas the latter, when the distance to the ball is lower than 7 meters.

goto_defaultpos() It moves players to the default position according to the invoked *schema*.

The *schema* depends on the ball position. This means that for an attacking team the player's default positions change with the ball position, so that the team itself is lined-up in the right manner to perform a coordinated attack action.

pressing() Activated in defence playmode, it allows the coordination of a team when the problem to be solved is *who must chase the ball?*.

We solved this problem via a negotiation based on visual and auditory information. The selected player makes pressing, chasing the ball that is in possess of the opponents.

defence() Activated in defence playmode, it provides the defenders, the wings and the midfielder with the skill of controlling and breaking the enemy's play.

This task is realized in two ways:

- 1. wings and midfielder: the players move dynamically between the ball and the enemies, to interdict the play. A mechanism select which one, among our players, has to take care of each enemy.
- 2. defenders: the outsiders dynamically keep a position between the enemy forwards and the nearest goalpost, to avoid a direct shoot to goal. The insiders keep symmetrically a position that allows to control an approaching central enemy, avoiding field areas to be not defended.

line-up() It is a defensive behavior for forwards and moves themselves to a default position.

smarcamento() It is an attacking behavior for the players without ball. It moves dynamically the players toward the most free area, preparing them to receive a passage. It is based on a random search algorithm that looks for a minimum of a function in a 2-D space. This function represents the degree of freedom of the player in the field.