Mathware & Soft Computing 8 (2001) 153-178

A Multi-agent System Based on Fuzzy Logic Applied to RoboCup's Environment *

E. Aguirre, J. C. Gámez, A. González
Depto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingeniería Informática. Universidad de Granada
18071-Granada
e-mail: {eaguirre,A.Gonzalez}@decsai.ugr.es, jcgamezg@terra.es

Abstract

Artificial Intelligent has been applied successfully to many and varied domains. In particular, the recent steps forward which are been producing in the development of dynamic, collaborative, real time and adversarial multi-agent environments are very interesting. In this work, we carry out a proposal, which uses a soccer robot simulator, called TeamBots, that simulates the "RoboCup Small", involved into the "RoboCup World Championship". The objective is to bring a team into operation, even though, this team initially only has available the basic behaviours needed to be operative and able to participate in a championship with other teams. So, we are proposed ourselves the development of the complete team, and subsequently in the future to go improving its capabilities and possibilities. In this work, we show the team's design in broad outlines, and we explain in detail the strategy of game for the different players, displaying moreover a part of the experimentation that has been carried out to state the correct operation of the team.

1 Introduction

Artificial Intelligent, and particularly the called Intelligent Systems, has had a notable progress a part from its application in different actual problems. In this work we show a application of *Artificial Intelligent* and *Robotics* to a facet of life which has been solely human up to date, such as the football. The initial idea arises from a congress in Tokio, in 1.993, in which emerges a project called *Robot J-League*, whose name changes to be *Robot World Cup Initiative*, more well-know as *RoboCup*. According to official web, *"RoboCup is an international joint project to promote AI, robotics, and related field"* [9]. Due to it, they use football as a tool for several and various purposes. Into the technical aspect, RoboCup is a system which links some characteristics very difficult of integrating in other different domains. We can

*This work has been supported by the CICYT under Project TAP1999-0535-C02-01

emphasize some of these characteristics such as dynamic, real time and distributed environment. Into the social aspect, RoboCup can be seen as a friendship and bright form of encouraging the investigation in these different areas, so as junior researcher as the researches with more experience, having as an objective a team's development which, in an intelligent way, can play football.

Robot World Cup Initiative (RoboCup) is an educational and research international initiative. It would be able to be considered as a standard problem in which a large range of technologies can be integrated and examined so as projects oriented to education. The main activities of RoboCup deal with different issues from technical conferences to develop of infrastructure [1], through educational programs and championship of robot, in which we can find leagues of simulators [15] , leagues of small robots [14, 19], leagues of legged robots [21], etc.

So RoboCup, as the several simultaneously celebrated conferences, has selected as primary domain a championship of football among teams of robots, with the objective of evaluating periodically the state of the art in principles of design of autonomous agents, multi-agents collaboration, acquisition of strategies, real time reasoning, robotics, and so on. Therefore RoboCup represents challenges in matter from match teams of robots in dynamic environment, to a platform oriented to research of software RoboCup itself.

RoboCup provides a challenging domain of investigation, which involves multiples agents which need to collaborate in on environment with adversary to reach a specific objective. It gives us the possibility of working in the study of Multi-Agent Systems in complex domains of real time, which require agents to operate efficiently as autonomously as on the part of a team. The team is formed by a system of multi-agent robots, with global perception and, actions and knowledge distributed.

The main objective is to obtain that a team formed by five robots can develop a game successfully. To get this purpose, it has been necessary to carry out the design of the skills of each kind of agent and to set up the necessary collaboration among them for they can make up a whole team. Besides, we have carry out the tasks of necessary implementation to get an operative team and prepared one to play against other existing teams. Moreover, our design makes possible, with certain easiness, the continuous improvement of the team and the incorporation of new ideas and methods that we could decide to put into practice as a result of our experimentation.

The result and conclusions of this study are very applicable in many domains [12] such as *Complex Systems (COSY)*. These systems can be integrated into Distributed Systems such as nets of distribution of energy or material, great floor of production with several stations and dynamics process or discrete ones with a large number of variables. Similarly, other field of application in which we can observe a direct extrapolation is the rescue of persons in extreme scenery [9], where the situation can be resolved without any necessity of endangering the life of the other person.

Likewise, it is very interesting by its properties of complexity, dynamics, imprecise knowledge and variables goals, whose principal features are real time, noisy, collaborative and adversarial.

In the same way, it is qualified as one of the best test beds, since it can be used for evaluating different techniques of multi-agent systems directly, according to the development and results of the game of team which are been implemented using these techniques. Among these techniques, we can emphasize the dynamic change of the roles or change of roles in simulation [17], prediction in real time [20], analysis of opposite team, learning of team [15, 16], precompiled plan [5, 13], predictive memory [3], and so on.

In order to build our team, it must be studied in a systematic way aspects such as: principles of design of autonomous agents, multi-agent collaboration, acquisition of strategies, reasoning in real time, etc. In particular, in our case, we have developed a team constituted by five robots: one goalkeeper, two defender players and two forward players; in this team, each robot is an autonomous agent with collaboration among partners in the situations in which it is required. The autonomy of each robot implies that each player develops his individual skills independently. These skills are developed by all kind of player with small nuances which help to each player to involve the activity more similar to its own kind. In the develop of these individual skills, we use basic techniques such as the geometry in order to obtain the angles, distances, and speeds needed for control of the robot in the domain. The collaborative skills are carry out among equals, in this team there is not any hierarchy, considering all of them exactly equals, that is, there is not any captain either preferences or privileges for no player's actions or decisions. Thus the team's strategy emerges of the collaboration of the different agents. In order to agree in these collaborative skills, we use the possibility of communication that the simulator provides.

Finally, for the development of the team, we use a system [2] based on fuzzy logic [22] which implements a whole strategy of game. The system of fuzzy logic lets to select the skills, the actions or decisions to take in each situation, and it will be detailed along this work.

At this moment, the problem has been situated and we have comment the techniques to use, in the following section we show a general description of the main ideas that we have considered at the design of our team, expounding some skills implemented for the players. In the section three we show the different kinds of agents and their behaviours. To finish, we analyze the experimentation done with our team and some notes about possible future works.

2 Overall Architecture

In this section we describe the philosophy that guides the design of the team. In the first place, by situating us, we display some definitions about agents, multi-agent system, distributed artificial intelligent, and a short taxonomy about multi-agent systems. In the second place, we show the simulator used for the development of our team and its experimentation. Finally, we show the own team, emphasizing the scalability of the design and some actions showed as an example. Moreover, we display the technique used for help to implement the skills and to take the suitable decisions while the game is playing.

2.1 Some definitions

For the development of our proposal, in the first place, we display some definitions and a taxonomy about multi-agent systems.

In this work, we are going to consider that *"an* agent *is an entity with perceptions, goals, actions and knowledge of domains, situated in an environment"* [18]. The way it acts is called its "behaviour". A group of agents in a Multi-Agent System which share common goals is said that they composite a "team". The members of a team or partners have to coordinate their behaviours to obtain compatible processes. Other agents of environment have opposite goals, they are the "adversaries" o "members of opposite team".

The *Distributed Artificial Intelligent* is considered as a sub field of *Artificial Intelligent* two decades ago. Traditionally, Distributed Artificial Intelligent was divided in two sub disciplines: *Distributed Problem Solving* and *Multi-Agent System.*

Multi-Agent Systems can be classified by:

- *Homogeneous non-communicating multi-agent systems*: all the agents have the same internal structure, including goals, knowledge of domain and possible actions. There is not possibility of communication.
- *Heterogeneous non-communicating multi-agent systems*: there are different options, from different goals, to have different models of domains and actions. There is not possibility of communication.
- *Heterogeneous communicative multi-agent system*: the comment is the same than the previous one, but now there is possibility of communication.

The last kind of multi-agent systems is the used by us in this work.

2.2 The Simulator of RoboCup "TeamBots"

From the beginning of international project RoboCup, it has had simulators in different languages [8, 10]: Lisp, C++, Visual C++, Java, etc, and in different platforms: Linux, Windows, UNIX, Iris, etc. We are chosen the TeamBots simulator [11], which is defined by its author as: *"TeamBots is a Java-based collection of application programs and Java packages for multi-agent mobile robotics research. The TeamBots distribution is a full source-code release. The simulation environment is written entirely in Java".* Moreover, by being developed in Java, it can be used in many platforms, whenever there is a compiler and an interpreter Java for this platform.

TeamBots can be used in many different domains among them, we show up SoccerBots, because it simulates the dynamic and dimensions of a game of RoboCup Small league, with two teams of five robots which compete on a field with similar dimensions to a ping-pong table, driving and shutting an orange ball, like a golf ball, to the opposite goal. The figure 1 displays scenery of SoccerBots.

In order to do easier the reading and comprehension of this paper, we are going to show the representation of the field by means of the simulator. The field of

Figure 1: Example of SoccerBots in TeamBots simulator

football is geocoding, that is, each point of field has its geographic coordinate. This geocoding is done in two equivalent systems of bidimensional coordinates. On one hand, we have Cartesian system of coordinate with values X, Y; and on the other hand, we have Polar system of coordinate with angle Θ and distance τ. The centre of these systems of coordinate is sited in the centre of the field.

The field of right side is the east field, and the field of left side is the west field. Respect to Cartesian coordinates, all the objects situated in east field have positive X coordinate, and negative X coordinate in the opposite field. Similarly all the objects in upper side of centre of the field have positive Y coordinate and negative Y coordinate in the opposite side. Respect to Polar coordinates, the centre of the field is the origin of coordinates, so all the objects, which are not in this point, have a major than zero τ distance. The angle Θ has a zero value in the horizontal with sense to east field, and this angle grows in the opposite sense to the hands of the clock. The limit values, which even not reached by the robots, are (-)1.47 meters in X-axis and (-)0.86 meters in Y-axis. Moreover, the angles are showed in radians since it is the international standard metric for angles. This information is displayed in figure 2.

Regarding the use of the distribution as test bed, we comment that is a software developed in Java, for this, it uses the Object Oriented Paradigm. The distribution is composed by a set of nine packages, among them "EDU.gatech.cc.is.abstractrobot" is the most interesting package for the control of robot, integrating a large diagram with approximately 38 items between classes and interfaces. In this diagram we find the class ControlSystemSS. The class that governs the system o ControlSystem, must inherit of ControlSystemSS and it must implement the "configure" and "takeStep" methods, which take charge of initialization and manage of the simulation. In the next section we explain briefly the diagram of classes implemented.

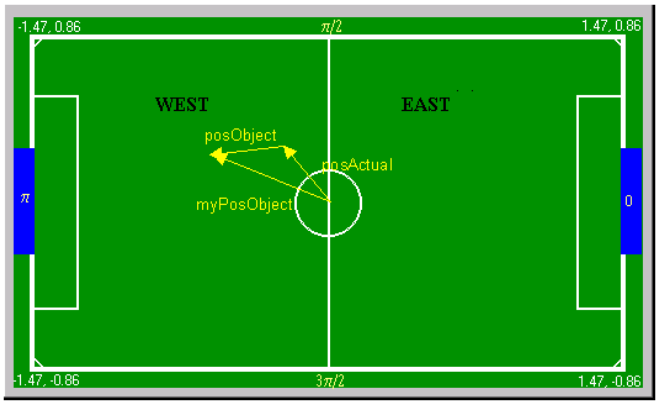

Figure 2: Representation of field in simulator

2.3 The team of agents

There are different ways of approach this work, which can be considered as different levels of trouble, for instance, each player can be considered independent and he can play a role or another in function of position that he has. Another option is that each player be independent but there is a communication with the others to develop any kind of game. A hierarchy could exist with a captain which organizes the game and the rest could carry out all the actions as the captain says. Regarding the different approaches, in our team there is not any hierarchy among players and in order to agree in the collaborative skills, the possibility of communication, that the simulator provides, is used.

In the same way, we have to take into account that the robots must have the capacity of going to the ball, driving the ball from a point to another, going to the opposite player, shutting, dribbling, defending, ..., that is, they must have the capacity of development the individual behaviours such as: moving the player along the field, avoiding the collisions with others, driving the ball, dribbling with the ball to an obstacle, defending the goal, so as the goalkeeper as by the others players; and the main collaborative skill: the pass of the ball to a partner.

To implement all these behaviours, we use the facilities that an Object Orient language like Java let us. In the first place, to carry out the development of the team we base on the behaviours which must have every player of football. For this, we create a class called PLAYER which implements all general behaviours of a player of football, and from this class we create the different kind of players with techniques of inheritance, overload and developing new behaviours. For example, we have a class PLAYER which develops all general behaviours of a player. As we know, a goalkeeper is a player, due to this, we create a class GOALKEEPER which inherits all behaviours of general player PLAYER, but for example, we refine the defense since it is different in a goalkeeper and we create exclusive behaviours of a goalkeeper. Furthermore, we can have different kinds of goalkeeper into the kind

of player GOALKEEPER, such as, a goalkeeper more risky which goes away from area; a goalkeeper which does not go away from area, and so on. Each one inherits his behaviour of general goalkeeper GOALKEEPER and we modify or refine some determined features, and so we can continue to the level which we wish to reach. This reasoning is similarly extrapolable to the others kinds of players. To explain this the figure 3 displays the generic diagram of classes.

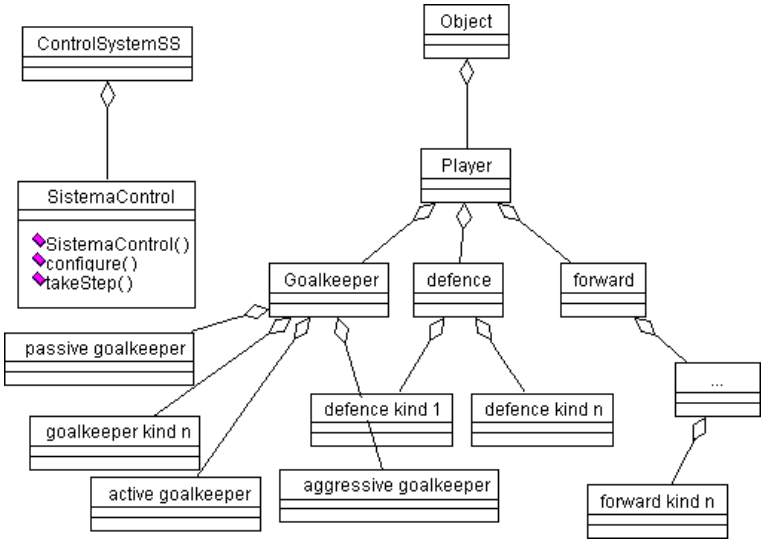

Figure 3: Example of generic diagram of classes

In order to develop the logic of reasoning, which governs the team's behaviour, an abstract class is introduced between the class general PLAYER and the determined kind of player. So we consider that the team's strategy of game is defined a part from the players' behaviour, but we can consider some variations of general strategy in each sub kind of the corresponding kind of player.

Finally, the diagram of classes of the team that we have implemented is displayed in figure 4.

So we have developed an architecture of behaviours composed of three levels, as is shown in figure 5.

In the top level, we have developed the more complex actions, which involve other simpler actions. These behaviours are the actions of the player in each situation in function of what logic of reasoning says in that situation. The list of these behaviours is:

- goTo, the player go to a position and avoid obstacles,
- goToNoStop, the player go to a position without avoiding obstacles,
- drive, the player drive the ball to a position,
- dribbling, the player drive the ball to a position and avoid objects,

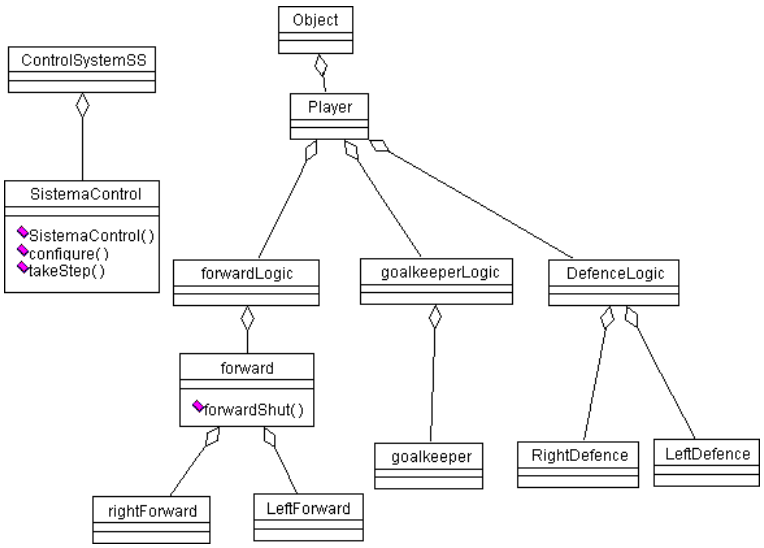

Figure 4: Implemented diagram of classes (without attributes and members)

- shut, the player shut the ball to a position,
- shutForward, the player shut the ball to a position without test the objects in the trajectory
- pass, the player pass the ball to a partner,
- receive, the player receive the ball from a partner,
- defend, the player defend the goal,
- clear, the player send the ball far away from the goal,
- shakeOff (one's attacker), the player look for a position free of opponent players.

In the low level, we have the functions that are used by the top level behaviour. This level is the intermediate level between top level behaviour and the level for the primitives of the simulator. This behaviour use the primitives, which the simulator provides, to help to behaviours of the top level so that they develop their actions. The list of these behaviours is:

- getDataSensor, get the data of sensors,
- calculateObstacleNearest, get the nearest obstacle from a point,
- restrictMovementArea, restrict the movement of the player to a defined area,
- avoidCollision, avoid the collision with obstacles,

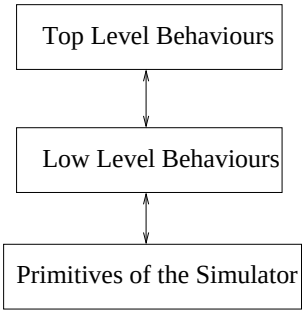

Figure 5: Levels of architecture of behaviours

- representField, it carry out the 3-D representation of the field,
- levelField, say us the level of the field between two points,
- trajectory, calculate the trajectory of the ball,
- senMessage, to help to pass-receive behaviours,
- receiveMessage, to help to pass-receive behaviours.

Finally, in the level of the primitives of the simulator, we found the functions that the simulator provides to control the robots.

In the next subsections, we are going to explain the avoidCollision behaviour as a example of the low level behaviours; similarly, we are going to show how we carry out the 3-D representation of the field. Besides, we are going to comment top level behaviours, which are concern with the interaction with the ball and we center the attention in the dribbling. So finally, we are going to explain the par excellence collaborative action, such as, the pass.

2.3.1 Avoid the collision

The first thing that we must do is to obtain all the information that the robot needs in order to know his situation and his environment. Once that the robot knows his environment, we can pass to the phase of movement. In this phase, the main thing that we bring out is the movement to avoid a collision with objects. In order to carry out this action, we calculate the robot's forward and the angle that the robot is able to turn in each simulation step. So we deduce the distance which the robot must check the obstacle to and the change of trajectory with a part of this information and other one such as the diameter of robot. In order to develop this change of trajectory we calculate the normal to the obstacle and the perpendicular to this normal. We calculate the sense of this perpendicular so that the robot turn away in a minor degree of his trajectory and so the robot goes to target following the new trajectory. This movement can be seen in the figure 6.

Figure 6: Movement of avoid the collision

2.3.2 3-D representation of the field

In this section, we are going to comment the development of a 3-D representation of the field in order to facilitate principally the decisions to shut, to pass and other behaviours which concern with the ball and the position of the players. In this representation, we divide the field in cells and we put each robot in the corresponding one. The position of robot is weigh up with a determined value according to whether the robot is a partner or an opponent and we decrease this value in proportion of the distance to the robot. So, we create a mountain and the top is the position of the rest of the robots. Besides, if there are more than one robot together, these values are added and we build a larger mountain and there is more than one top. Each robot develops his own representation in order to carry out his own behaviours, decisions and purposes. So, each representation belongs to the robot self, it is relative to the robot and this representation is different one of the other robots. By finalizing this representation, if we extrapolate this 3-D representation, the field of football converts itself in a set of mountains and valleys, where the mountains mean difficult places to cross away and valleys mean easy place to access, and so we can decide if the possible trajectory of ball is obstructed or not and the degree of obstruction. We can observe it in the figure 7.

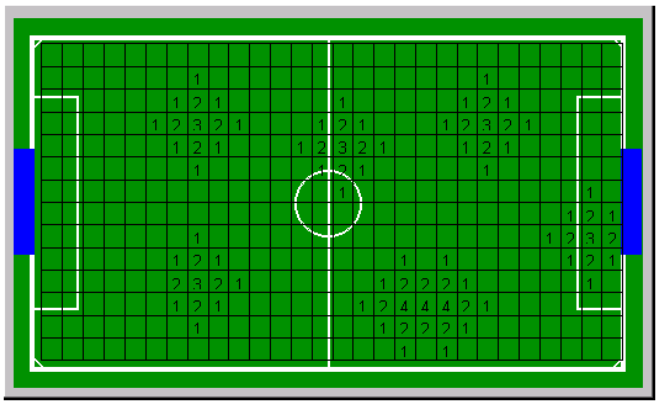

Figure 7: Representation of the field

2.3.3 Interaction with the ball

As we know, the main skill that a robot must have is the interaction with the ball. Particularly the robot must be capable of driving the ball from a point of field to other. In addition to this kind of interaction, the robot must know the way of shutting the ball to a target. The defence of the goal can be considered as other kind of interaction, in that the robot calculates the trajectory of the ball and he goes to a point in this trajectory in which he can get the ball to perform other action later.

The dribbling is one of the more difficult behaviours into the set of interaction with the ball. In this case, we enhance that the robots do not have got members. So for the simulator, the robots can be considered as spheres, taking into account the difficult process which a bigger sphere controls the movements of other smaller one.

In short, the explication of this movement could be: we calculate three vectors, one vector from ball to destination, other vector from player to ball and other one from player to destination. In function of these vectors we carry out a series of computing to accomplish the dribbling. With broad strokes, the following is done: we test if the opposite player is far away and in this way we do nothing. In other case, we calculate the position of dribbling which the player must go to: in first place, the player goes to the obstacle, striking the ball in this movement and going far away it from the obstacle. Immediately next the player goes to the ball and drives it to the wished position. Figure 8 displays this movement.

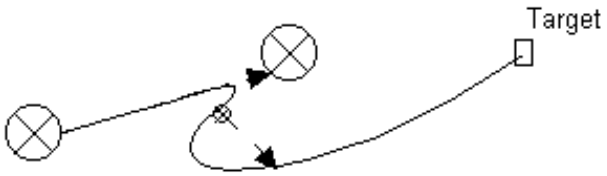

Figure 8: Movement of dribbling

We have considered that the defence of the goal by means of the goalkeeper is different to the defence of the goal by means of other players. In this case, the goalkeeper moves himself in a parallel line to the goal, whereas the ball is far away. When the ball comes near to the goal, the goalkeeper acts in a different way according to his implemented logic, which we describe later.

2.3.4 Collaborative skill (pass to partner)

Now we are going to comment the pass from point of view of collaborative action between robots. For this, although it is not strictly necessary, some systems carry out a previous communication to develop this skill. In our proposal we set up a

communication between these players, to make easy the transmission of the purpose, and so they can reach a consensus to make the pass. Therefore, the first thing that we have to do is the implementation of the functions that let them this communication, enabling the communication not only with an only player, but also with all players at the same time. Immediately next, we implement the necessary for the pass the ball from one player to another.

This process of pass can be synthesized in the fulfillment of two complementary actions. On the one hand the player which carries out the pass must communicate with the receptor and bring the ball closer to him. On the other hand, the player that receives the ball must have the capability of receiving the partner's communication and must agree with the first one to receive the ball.

We decide to implement it in two functions: one for the pass of ball and other by receiving the ball, which is helped by the function of communication.

3 Description of the different kinds of considered agent

After briefly commenting the team, we are going to explain the development of fuzzy logic used in the definition of the behaviour of each particular player.

There are multiples alternatives to chose the player's behaviour, which he is going to realize in each situation [4, 6, 7, 23]. Here we have used fuzzy logic [22], but with some differences according to kind of player which will carry out the action, since, as we said before, the game of a goalkeeper is different to the game of a forward player. This difference is implemented with different classes for the logic of each kind of player.

As we know, this separation or difference can be done to different levels having to reach a commitment among efficiency, maintenance and functionality. We have chosen the alternative that we are going to explain bellow.

3.1 The specific observable behaviour of the goalkeeper

We have implemented a goalkeeper with the peculiarity of having a "low-risk" behaviour. Here, we want to manifest that the goalkeeper does not go out of its area in front of any dangerous situation, but the goalkeeper defends the goal very close to it. However, this feature does not imply any limit because we can modify the attitude with correction of some parameters or introducing weights to weight up the election of rules which are activated, or others different methods.

The logic, which governs the observable behaviour of a goalkeeper, could be defined as we show bellow. So, we considerate the goalkeeper's point of view:

- If I am out of my area, it means that I am for away of my goal and the place that I must defend. Then I must go to the position by defect, the goal.
- If the ball is out of my area or "action area", and besides the previous case is false, I must go to the position of defense of my goal.

- If I am in my area, the ball is in my area too and the ball is in front of me, then the situation is dangerous since the ball is near of my goal, I must go to ball to clear, I will shut it far away.
- If I am in my area, the ball is in my area too, and the ball is at the same latitude than myself then I must go to ball and clear, but in this case is needed a different behaviour than in the previous case since this situation is more dangerous than the before one. I must move behind the ball and drive it to the opposite field and, when I can shut the ball, I will do it.
- If I am in my area, the ball is in my area too, and the ball is behind of me and I can score a goal in my own goal. Then I must locate the closest opposite player, and block him, since securely he has brought the ball behind me and I am not make any other thing, because if I go to the ball, I will score a goal.
- If I am in my area, the ball is in my area too, and the ball is behind of me and I can not score a goal in my own goal, then I must go to the ball to recuperate it and clear.

The fuzzy variables that composite this logic are the next:

- AREA: in function of position of the player's distance to goal or the ball's distance to goal, it shows if the player or ball is in or out of area.
- BALLPOS: in function of the ball's position and the player's position, it shows if the ball is in front of, at the same latitude or behind of the player.
- SCORE: in function of position of goalkeeper and goal, it shows the possibility of the goalkeeper score a goal in its own goal.

The fuzzy set corresponding to these fuzzy variables are displayed in figures 9 and 10.

In fuzzy variable AREA, we observe that up to a distance lower than 0.4 meter we considerate that the player is into the area.

From this point, the fuzzy zone begins up to 0.6 meter. From 0.6 on upper values we considerate that the player is out the area. These values are bounded due to dimensions of field of game, since, for example, the middle of the field is 1.5 x 1.5 meters approximately, so that these values represent a 1/3 of middle of the field.

In fuzzy variable BALLPOS, the choice of player's position regard to ball, (in front of, at same latitude, behind), is done in function of player and ball's values in x-axis. For example, in the east field, if the x value of player is upper than x value of ball, the ball is in front of player. This choice must be independent of the field in that we play, for this, we eliminate the signs (positive, negative) by using the multiplication operation. We make player's x-value squared and we multiply player and ball's x-values, so if player's x-value squared (B) is upper than player's x-value times ball's x-value (A), then the ball is in front of the player, and so on. Figure 11 is shown an example for the best understanding of this variable. Also

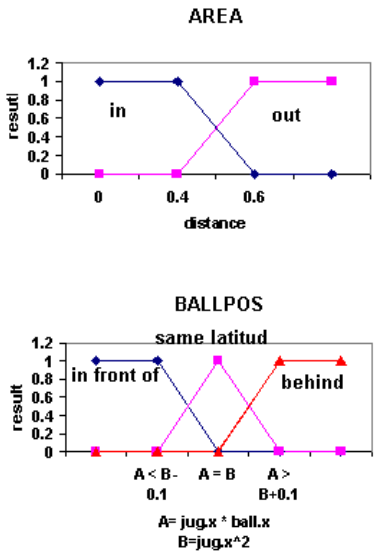

Figure 9: AREA on the left and BALLPOS on the right

we want to comment that the value of this fuzzy variable is only considered when the ball and player are near and they are in the same area.

Furthermore, we considerate 0.1 meter, since the player's diameter is 0.12 meters approximately and the ball's diameter is 0.05 meters approximately. So we considerate that reliance rate is lower than the necessary for a player.

By means of the fuzzy variable SCORE, we pretend to test if the goalkeeper can score a goal in own goal. For this we take the vector of the ball's position and the vector of the goalkeeper's position, then we subtract the position of the goal to each one of this vectors. Finally, in function of the angle between these result vectors and of a reliance rate, it shows the possibility of scoring a goal by the goalkeeper. We considerate a value of 0.05 since the angles are taken in radians and this value corresponds to an angle very little but it is sufficient to go to goal and to strike the ball, so the ball changes its trajectory turning off the goal. We display two examples of these calculates in figure 12.

The fuzzy rules which govern goalkeeper's observable behaviour corresponding to fuzzy variables and sets that we explained above, are the following:

1. If Area(pos, out) $--> $ goTo(defaultPos)
2. If Area(ballPos, out) $--> $ defend()
3. If Area(pos, in) y Area(ballPos, in) y BallPos(inFrontOf) $--> $ clear()

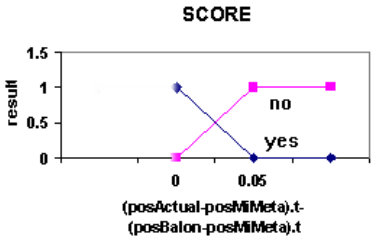

Figure 10: SCORE

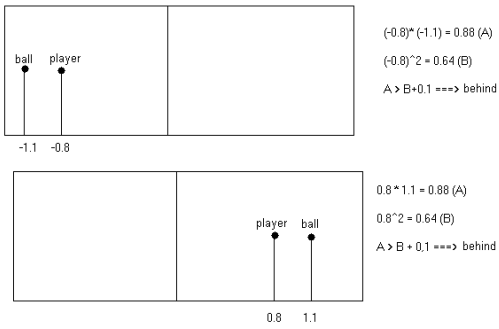

Figure 11: Example to explain the variable BALLPOS

4. If Area(pos, in) y Area(ballPos, in) y BallPos(sameLatitud) $--> $ dribbling()
5. If Area(pos, in) y Area(ballPos, in) y BallPos(behind) y Score(yes) $--> $ goToNoStop(posClosestOpposite)
6. If Area(pos, in) y Area(ballPos, in) y BallPos(behind) y Score(no) $--> $ goTo(BallPos)

As a note of implementation, we can comment that in case of dead heat among rules we take the first rule in the displayed order.

This is the logic's basic description, fuzzy in this case, which makes from a robot, a goalkeeper with reasoning.

3.2 The specific observable behaviour of the forward player

Regarding to forward player, it can be said that we have implemented a "low-risk" forward player too, since the player does not go to the ball in any position, but the player wait to the ball comes into his action area.

In this case, the logic, which governs the observable behaviour of a forward, could be defined as we show below. So we considerate the forward's point of view:

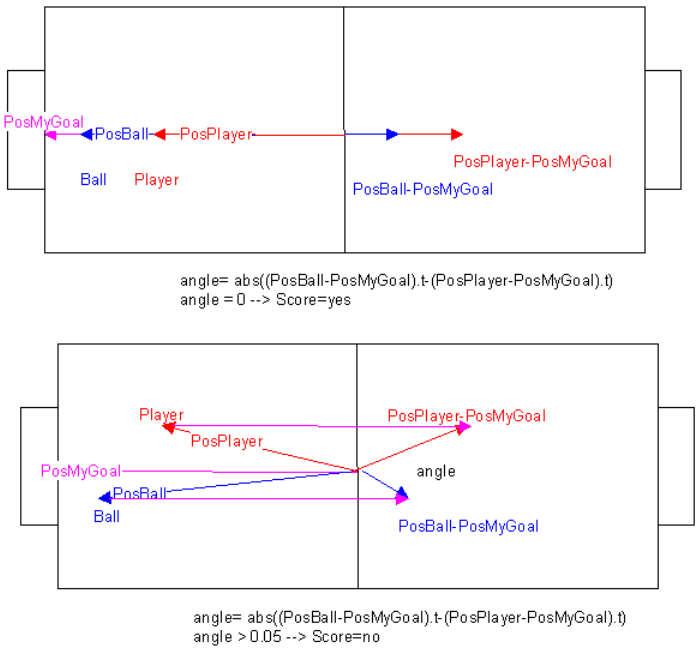

Figure 12: Example to explain the variable SCORE

- If the ball is in my field, I am out of my area and I have not got the ball, then I can not do anything, so I must come to my default position, since I can be needed in that position which is near to the centre of the field.

- If the ball is in my field, I am out of my area but I have got the ball, then I try to drive the ball to the opposite field and I try to put the ball far away of my goal, so that I will shut the ball to the opposite goal when I can do it.

- If the ball is in my field, I am in my area and the ball is not far away, then I try to get the ball and drive to the opposite goal.

- If the ball is in my field, I am in my area and the ball is far away, then there are not possibility of I getting the ball, so I must stay in my default position.

- If the ball is in the opposite field and I cover the opposite goal, then securely any partner would have got the ball and he would try to shut to goal, so I must look for a free position and clear the possible trajectory of ball.

- If the ball is in the opposite field, I do not cover the opposite goal and the goal is free, then I must get the ball and I must try to shut the ball to opposite goal.

- If the ball is in the opposite field, I do not cover the opposite goal, the goal is not free and there is a free partner in front of me, then I must pass the ball to the partner.
- If the ball is in the opposite field, I do not cover the opposite goal, the goal is not free and there is not a free partner in front of me, then I must try to score a goal, so I must drive the ball to the opposite goal.

The fuzzy variables that composite this logic are:

- AREA: It is the same variable that in the goalkeeper case.
- BALL: in function of the ball's position and the player's position, it shows if the player has got the ball, the ball is near to the player or the ball is far away to the player.
- BALLFIELD: in function of the ball's position, it shows if the ball is in my field or in opposite field.
- GOAL: in function of the level of the trajectory from the ball to goal in 3D-representation, it shows if the goal is free to shut.
- GOALCOVERED: in function the player's position and the ball's position and the angle between them, it shows if I cover the goal. It is useful to check if I hold up to the shut of a partner.
- FORWARDPARTNER: this variable looks for a partner more forward player than me and it shows if he is free.

The fuzzy sets corresponding to these fuzzy variables are displayed in figures 13 and 14 whereas the fuzzy sets regarding to AREA were showed in previous subsection.

In fuzzy variable BALL, we observe that the player has got the ball at a distance lower than 0.1 meter. For values between 0.15 and 0.3 meters, the player is near to the ball and for values upper than 0.35 the ball is far away from player. The values 0.1 and 0.15 are considered since the robot's diameter is approximately 0.12, so that at a lower distance we considerate that the robot has got the ball, and at an upper distance we considerate that other robot can be situated between the player and the ball. The value 0.3 is approximately 1/5 of the middle of the field and the ball runs more speedily than the robot, so that an upper distance than 0.35 we considerate that the ball is far away.

Regarding to the fuzzy variable BALLFIELD, the robot has the capability of playing in the east or west field. In order to obtain this, we have to eliminate his dependency of the field. This independence is obtained thanks to this fuzzy set and the variable "field", which takes the value "1" whether the team plays in west field or "-1" in opposite case. This fuzzy variable is shown in figure 13. We take the 0.1 value since the diameter of the player is approximately 0.12 meter.

In the fuzzy variable GOAL, we use the 3D-representation of the field. We trace the trajectory from the ball to goal, according to the zones that the trajectory

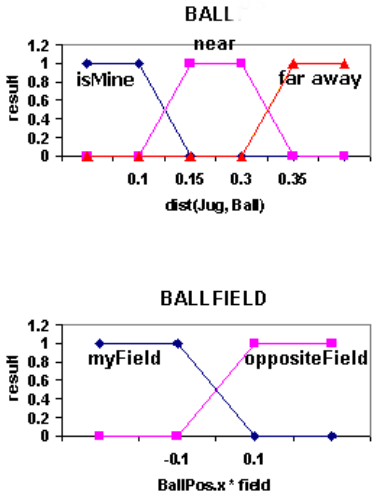

Figure 13: BALL on the left and BALLFIELD on the right

crosses away and the levels of this zones, the fuzzy variable shows if the goal is free or not.

In the fuzzy variable GOALCOVERED, we create two vectors, one from the player to the opposite goal and other from the ball to the opposite goal. According to the angle that these vectors composite it, the fuzzy variable shows if the player covers the goal. The values of this set are taken in an experimental way and they are checked with simulations.

The fuzzy variable FORWARDPARTNER is similar to the GOAL fuzzy variable.

The fuzzy rules, which govern forward's observable behaviour corresponding to fuzzy variables and sets that we explained above, are the following:

1. If BallField(myField) and Area(out) and Ball(isMine) $->$ goTo(defaultPos)
2. If BallField(myField) and Area(out) and Ball(NOTisMine) $->$ shut(oppositeGoal)
3. If BallField(myField) and Area(in) and Ball(farAway) $->$ drive(oppositeGoal)
4. If BallField(myField) and Area(in) and Ball(NOTfarAway) $->$ goTo(defaultPos)
5. If BallField(oppositeField) and GoalCovered(Yes) $->$ ShakeOff(posFree)
6. If BallField(oppositeField) and GoalCovered(No) and Goal(free) $->$ shut-Forward(pos)

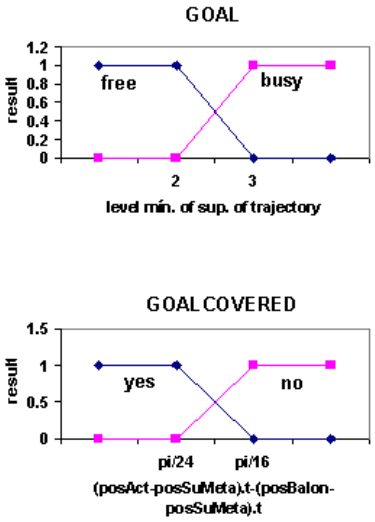

Figure 14: GOAL on the left and GOALCOVERED on the right

7. If BallField(oppositeField) and GoalCovered(No) and Goal(busy) and ForwardPartner(free, pos) $->$ pass(pos)
8. If BallField(oppositeField) and GoalCovered(No) and Goal(busy) and ForwardPartner(busy, pos) $->$ drive(oppositeGoal)

3.3 The specific observable behaviour of the defence player

In this section, we are going to describe the logic of a defence player. We have thought of a defence player as a mixture between forward player and goalkeeper's behaviours, so that we have developed a "low-risk" defence player. So, finally we have created a "low-risk" team. We say that our team is "low-risk" in the sense that they do not move to the ball continuously, but that they wait for the ball into in their action area. In the same way that the above kinds of logics, we can softly modify his behaviour either by means of the correction of some parameters or introducing weights to weight up the election of rules or other different methods.

In this case, the logic, which governs the defence player's observable behaviour, could be defined as we show bellow. So we considerate the defence's point of view:

- If I have the ball and there is a partner in front of me, which is free, then I must pass the ball to the partner.
- If I have the ball and there is a partner in front of me, which is not free, then I must avoid the dangerous situation and I must clear the ball to opposite

goal, so that other partner can get the ball, drive it to opposite goal and score.

- If the ball is in my field, the ball is not in front of me and I can score a goal in my own goal, then I must locate the closest opposite player and block him, so that other partner can get the ball and drive to opposite goal.
- If the ball is in my field, the I have not the ball and the opposite player is not far away, then I must go to a strategic defensive position to prevent a possible attack from opposite team.
- If the ball is in my field, the I have not ball and the opposite player is far away, then the situation is not dangerous, so that we can go to the ball and begin a new move for play.
- If the ball is in opposite field, and I have not got the ball, then I must go to my default position to avoid strike against.

For this logic, we take from goalkeeper the fuzzy sets and variables: SCORE, BALLPOS; and we take from forward player the fuzzy sets and variables: BALLFIELD, FORWARDPARTNER, BALL, and OPPOSITEBALL which is quite similar to BALL

So the fuzzy rules, which govern defence's observable behaviour, are the following:

1. If Ball(isMine) and ForwardPartner(free) $->$ pass(pos)
2. If Ball(isMine) and ForwardPartner(NOTfree) $->$ clear()
3. If BallField(myField) and BallPos(forward) and Score(Yes) $->$ goToNoStop(posClosestOpposite)
4. If BallField(myField) and Ball(isMine) and OppositeBall(farAway) $->$ defend()
5. If BallField(myField) and Ball(isMine) and OppositeBall(NOTfarAway) $->$ goToNoStop(Ball)
6. If BallField(oppositeField) and Ball(NOTisMine) $->$ goTo(defaultPos)

When we have described some of the more important components to understand the design of our proposal, now we are going to show the experimentation with our team.

4 Experimentation

In this section we are going to display the result of simulations by means of tables and then we will show the conclusions of development of the team.

The developed team has been carried out, as all software, as a result of a evolution through different versions, with a continuous improvement from a version to another. We do not display all results but we display only the test on the last version, which has implemented all behaviours so individual as collaborative and competitive, with representation 3-D of field and the use of fuzzy logic in the complete team.

The main features of our team (*UGRTeam*) are the following: our team is a low-risk one whose players go to the ball when it is near. Besides, the team has a well-defined strategy of game with determined position of the players. The players can communicate among them to develop a collaborative behaviour. In this version, the players have defined roles in design time, but in a later version, we think to develop the dynamic change of roles.

4.1 Opponent Teams

To elaborate this test we have used other teams that bring the distribution. These teams have been developed by groups, which contribute a team with its code and comments.

The teams can be classified mainly by two features. On the one hand, the teams can be classified in function of their players, that is to say, in function of whether the players are homogeneous or heterogeneous. So, we have the following classifications.

1. Homogeneous: BasicTeam, GoToBall, CommTeam, AIKHomo, BrianTeam, DoogHomoG, LoneForwardTeamHomoG, SchemaDemo, PermHomoG

2. Heterogeneous: DTeam, DaveHeteroG, DoogHeteroG, FemmeBotsHeteroG, MattiHetero, SibHeteroG, SchemaNewHetero, CDTeamHetero

3. The autor does not say us this feature: KeChZe, JunTeamHeteroG

On the other hand, we can classify the teams in function of the strategy of game, even though the teams are different ones, we bring out three kinds of strategy:

1. Simple Strategy, the teams which have simple strategy, non-well-defined players' position, or they are test teams: BasicTeam, GoToBall, CommTeam, BrianTeam, DaveHeteroG, SchemaDemo, SchemaNewHetero

2. More Elaborated Strategy, teams which have a more elaborated strategy, well-defined players' position or more intelligent behaviour: AIKHomo, CDTeamHetero, FemmeBotsHeteroG, KeChZe, LoneForwardTeamHomoG, MattiHetero, PermHomoG, SibHeteroG

3. Non-sport strategy, teams which have a non-sport strategy. The objective of these teams is setting back the player of the opposite team, so they have clear the field and they can score goals without opponents: DTeam, DoogHeteroG, DoogHomoG, JunTeamHeteroG

4.2 All against UGRTeam

This kind of experimentation that we are going to display enables us test the performance of team in a simulation in which our team must come up against the other teams, which have different behaviours and strategies. To carry out this experimentation, we have accomplished several simulations of matches against each team. When the simulations have finished, we calculate the difference between the number of goals that we have got and the number of goal that the opposite team has got in each match. When this value is positive it means the advantage obtained by our team while a negative value means that our team has lost the match by that number of goals. Actually the values that we are going to show, are the average values and their standard deviations obtained from several simulations. In this way, the average value shows us the performance of our team against the opponent one and the standard deviation value shows us the variability of this average value. So, a positive average value means that our team generally gets a victory against the opponent team and a negative average value means that our team generally get lost against the opponent team. The table 1 displays the averages values and their standard deviations that we have obtained from the simulations while table 2 shows the total average and its standard deviation.

Table 1: Table of results of UGRTeam against other teams

Teams	Average values	Standard Deviation
BasicTeam	7.4	3.16
GoToBall	4	2.04
CommTeam	8.6	3.74
AIKHomoG	-0.8	0.74
BrianTeam	5.1	2.42
CDTeamHetero	4.2	2.03
DTeam	0.3	0.45
DaveHeteroG	0.7	0.78
DoogHomoG	-1.2	1.6
DoogHeteroG	-1.2	1.16
FemmeBotsHeteroG	1	1.48
JunTeamHeteroG	7.2	2.82
Kechze	1	0.63
LoneForwardTeamHomoG	0.6	1.01
MattiHetero	-0.2	0.6
PermHomoG	0.9	0.94
SibHeteroG	2	1.54
SchemaDemo	2.1	1.51
SchemaNewHetero	0.1	0.94

Table 2: Total average

Average of average values	Standard deviation
2.2	2.93

Table 3: Results grouped by kind of opponent team

Kind of strategy	$\sum$ Average values of the kind	Average of the kind	Standard Deviation
Simple Strategy	29.9	4.27	2.73
More Elaborate Strategy	6.8	0.85	1.4
Non-sport Strategy	5.1	1.27	3.47

These results show that our team is only in trouble when it plays against four teams. Two of these teams belong to more elaborate strategy teams while the other two teams belong to non-sport strategy teams. In general terms we observe a good result since the total average is positive.

In order to ease the analysis of these data, we have classified the results in function of the strategy of the team, and the average value into each group has been calculated. The table 3 shows the results grouped by the different kind of strategy.

So, in the group of teams with simple strategy, we observe that our team wins to the opposite team by a average of 4.27 goals, that is to say, in general, we obtain a good advantage over this group. In the group of teams with a more elaborate strategy, we observe that our team generally gets a victory but with an advantage very small. This result is due to our team won against 6 teams but it lost against 2 one. In the group with non-sport strategy, we observe a similar case that the previous one, but in this case the advantage is better, getting 1.27 goals of average value. Here, we observe that our team won against 2 teams and it lost against 2 one, but we our team got a great advantage against one of them; due to this advantage we get this average value. This great variation is because of the strategy of these teams, since if the opponent team get to block our goalkeeper, they have clear our goal and they can score easily, but in this movement in order to block our players, they can clear their goal and thus our team can score easily too, that is to say, the result depend on how they block our players.

The result of the analysis of the grouped data is supporting the first conclusion, that is, our team wins with certain easiness when it plays against teams of simple strategy, while regarding to the teams of more elaborate strategy our team finds more difficulty to break them down and sometimes it can lose the match but obtaining a low difference between the own goals and the opponent ones. In the case of non-sport strategy teams the result depends on how the opponent team blocks our players since our team, in current version is not properly ready to play against

this kind of teams.

5 Conclusions and future works

In this work, we have put into operation a team with enough skills to be competitive to basic level, and it can serve as platform of test about different investigations and developments. It is important to put manifest the large flexibility of the proposal and its capacity of improvement for instance by refining some components of model.

By this way, this work pretends to be an initial contact with the topic in question, with the definition of a team that develops all skills of a player of football, with the intrinsic limitations of the environment. So we have carried out our initial proposal of development and implementation necessary so that a team of five robots can develop a match of football successfully.

In the team, the final game is the result of the collaboration of the different kinds of agent, which have got behaviours so general as particular. Thanks to is, we can modify easily the players' behaviour only modifying the general class. On the other hand, it is very easy to create agents of one determined kind of player but with different level of risk. For this, we only have to generate a new subclass of the concrete kind of player and modify lightly. All these features offer us many possibilities of change of the observable behaviour of the team, realizing small adjust in the suitable classes.

Finally, in the future works, it can be though the development and evaluation of different choices of used techniques in this work such as, the development of other variation or other fuzzy logic which governs the team; or the development of the things that you needed to the dynamic interchange of roles.

References

[1] S. Achim, P. Stone, and M. Veloso. Building a dedicated robotic soccer system. In *Proceedings of the IROS-96 Workshop on RoboCup*, pages 41–47, Osaka, Japan, November 1996.

[2] E. Aguirre, J.C. Gámez, and A. González. Un sistema multi-agente basado en lógica difusa aplicado al ámbito de la robocup. In *II Workshop Hispano-Luso de Agentes Físicos*, pages 131–145, Madrid, 2001.

[3] M. Bowling, P. Stone, and M. Veloso. Predictive memory for an inaccessible environment. In *Proceedings of the IROS-96 Workshop on RoboCup*, pages 28–34, Osaka, Japan, November 1996.

[4] G. Gutiérrez F. Fernández and J.M. Molina. Coordinación global basada en controladores locales reactivos en la robocup. In *Primer Workshop hispano luso de agentes fisicos*, pages 73–86, September 2000.

[5] R. Iso and H. Inazuni. A multi-layered planning architecture for soccer agent. In *Robocup-97: Robot Soccer World Cup I*, pages 513–518. H Kitano, 1997.

[6] M. Kamel and H. Ghenniwa. Coordination of distributed intelligent systems. In Aminzadeh and M. Jamshidi, editors, *Soft Computing: Fuzzy Logic, Neural networks and Distributed Artificial Intelligence*, pages 229–260. Prentice-Hall, 1994.

[7] V. Matellán. *ABC: Un Modelo para el Control de Robots Autónomos.* PhD thesis, Universidad Politécnica de Madrid, 1998.

[8] I. Noda. Soccer server: a simulator of robocup. In *Proceedings of AI Symposium'95*, pages 29–34. Japanese Society for AI, December 1995.

[9] The official RoboCup website. http://www.robocup.org.

[10] The official TeamBots Simulator website. http://www.cs.cmu.edu/~trb/TeamBots.

[11] The official TeamBots website maintenance by Tucker Balch. http://www.teambots.org.

[12] A. Oller, J. L. de la Rosa, R. García, J.A. Ramón, and A. Figueras. Micro-robots playing soccer games: A real implementation based on a multi-agent decision-making structure. *Intelligent Automation and Soft Computing. Special Issue on Soccer Robotics: Micro-Robot World Cup Soccer Tournament'97*, 6(1):65–74, 2000.

[13] P. Stone. *Layered Learning in Multi-Agent Systems.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, December 1998.

[14] P. Stone and M. Veloso. The CMUnited-97 simulator team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 387–397, Berlin, 1998. Springer Verlag.

[15] P. Stone and M. Veloso. Team-partitioned, opaque-transition reinforcement learning. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 261–272. Springer Verlag. Berlin, 1998.

[16] P. Stone and M. Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48(1):83–104, 1998.

[17] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[18] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[19] M. Veloso, M. Bowling, S. Achim, K. Han, and P. Stone. The CMUnited-98 champion small-robot team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 61–76. Springer Verlag, Berlin, 1999.

[20] M. Veloso, P. Stone, K. Han, and S. Achim. Prediction, behaviors, and collaboration in a team of robotic soccer agents. Technical report, Computer Science Department, Carnegie Mellon University, Pittsburgh, 1998.

[21] M. Veloso and W. Uther. The CMTrio-98 Sony legged robot team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98:Robot Soccer World Cup II*, pages 491–497, Berlin, 1999. Springer Verlag.

[22] L.A. Zadeh. The concept of linguistic variable and its applications to approximate reasoning. *Part I Information Sciences vol. 8, pages 199-249, Part II Information Sciences vol. 8, pages 301-357, Part III Information Sciences vol. 9, pages 43-80*, 1975.

[23] Y. Zhang and A.K. Mackworth. Using reactive deliberation for real-time control of soccer-playing robots. In *Robocup-97: RobotSoccer World Cup I*, pages 508–512, 1997.