COMPARING BEHAVIOR IN AGENT MODELLING TASK

José Antonio Iglesias, Agapito Ledezma and Araceli Sanchis
Universidad Carlos III de Madrid
Avda. de la Universidad, 30, 28911. Leganés (Madrid)
Spain
{jiglesia,ledezma,masm}@inf.uc3m.es

ABSTRACT

In multi-agent system, agents have to analyze several features in order to adapt their behavior to the current situation. This extracted information is usually related to the environment and other agents influence. In this paper we present a method that compare two different agent models in order to extract the qualitative differences between them. This proposed comparative method captures several features of the two agent models and model them considering its behavior.

KEYWORDS

Multi-agent Modelling, Coaching, RoboCup, Classification Algorithm.

1. INTRODUCTION

Agent modelling techniques have been used for various purposes. The idea of extract knowledge from other agents' behaviour was used, originally, in the field of the game theory [Turocy and Von Stengel, 2001]. After not considering information about opponents in game mechanics, it was realized that knowledge about the opponent's strategy can enhance the game results. Following this initial research, many others researches have been working to develop different ways of modelling other agents. Carmel and Markovitch [Carmel and Markovitch, 1996] proposed a method to infer a model of the opponent's strategy which is represented as a Deterministic Finite Automaton (DFA).

Perhaps, one of the most interesting environments where agent modelling has been used is the robotic soccer domain. The Robot World Cup Initiative (RoboCup) [Kitano *et al*., 1997a] is an international joint project to encourage AI, robotics and related fields. It provides a standard problem where many intelligent techniques must be integrated and examined. RoboCup chooses to use soccer game as a central topic of research, aiming at innovations to be applied for socially significant problems and industries [Kitano *et al.*, 1997b]. One of the challenges for the near future is opponent modelling, a research which can be used both RoboCup and general multi-agent system.

This research is driven by the *RoboCup Online Coach Competition* goal using formations recognition. The remainder of this paper is organized as follows: Section 2 presents the modelling environment used in this research. Section 3 describes the overview structure followed in this research. Section 4 provides a method to recognize formations and the results obtained are shown. Section 5 describes the proposed comparing method. Finally, conclusions and future works are drawn in section 6.

2. THE MODELLING ENVIRONMENT

In this section, we present the environment where our comparative method has been used. Also, the goal of this research is showed in section 2.3.

2.1 The RoboCup Simulation League

One of the leagues competitions in RoboCup is *Simulation league*. The *RoboCup Simulation League* uses

the Soccer Server System [Chen *et al.*, 2001] to simulate the field and the objects. Each player consists of a unique process that connects via a standard network protocol to the server. This server keeps track of the current state of the world, executes the actions requested by the clients, and periodically sends each agent noisy information about the world. There are 11 players (10 fielders and 1 goalkeeper) that can only perceive objects that are in their field of vision and both the visual information and the execution of the actions are noisy.

There are three competitions in this league: 2D, 3D and Coach. In this paper, *Online Coach Competition* goal is treated, but some aspects about *2D Competition* are considered, as well.

2.2 The RoboCup Simulation League

In the *RoboCup 2D Competition*, in order to perform a soccer game, technologies like multi-agent collaboration or strategy acquisition must be incorporated. There have been many papers and studies related to opponent modelling in this robotic soccer domain: Stone et al. [Stone and Veloso, 2000] introduce "ideal-model-based behaviour outcome prediction" (IMBBOP) which models the result of the other agents' future actions in relation to their optimal actions based on an ideal world model. But, this first work into automated opponent-modelling, assumes that the opponent plays optimally.

Ledezma et al. [*Ledezma et al.*, 2005] present an approach to modelling low-level behaviour of individual opponent agents, OMBO (Opponent Modelling Based on Observation). Riley et al. [Riley and Veloso, 2000] propose a classification of the current adversary into predefined adversary classes. Based on this work, Steffens [Steffens, 2002] presents a feature-based declarative opponent-modelling (FBDOM) which identify tactical moves of the opponent.

2.3 The RoboCup Online Coach Competition

The *RoboCup Coach Competition* mainly deals with an agent (coach) providing advices to another agents about how to act. This special agent can only support its team by giving messages to its player in a standard coach language called CLang [Chen *et al.*, 2001]. About the Coaching problem raised in *Coach Competition*, Riley et al. [*Riley et al.*, 2002a] present a general description of the coaching problem; this is the first step in understanding advice-based relationships between automated agents. One of the reasons in the research to consider a coach role in a team of autonomous agents is that a coach role provides a method of oversight for the agents and can aid in the creation of agents with adjustable autonomy [Scerri *et al.*, 2001]. Riley et al. [*Riley et al.*, 2002b] justify that coaching can help teams improve in simulated robotic soccer domain.

2.3.1 RoboCup 2005 Official Rules for the Online Coach Competition and Problem Definition

According to the RoboCup 2005 official rules for the *Coach Competition* [RoboCup Organizing Committee, 2005], the coach has to analyze logs of a given team and discover the play pattern of it. The term play pattern is used to describe a simple behaviour that a team performs which is predictable and exploitable for the coaches [Kuhlmann *et al.*, 2004]. For example, a possible pattern could be: "*The players form a 4-4-2 formation in which the 4 midfielders form two layers of 2 and 2 players respectively, forming an arrangement of 2-2-2-4"*.

The two phases required in this competition are as follow:

- **Offline analysis:** The coach analyzes the log-files of a team and has to detect specific pattern (simple behaviors) from the log files. Each coach has access to a log file with and without the pattern (*Pattern Log-File* and *No-Pattern Log-File*) and thus has to look for their qualitative differences to recognize the pattern correctly [RoboCup Organizing Committee, 2005]. Then, the coaches can create some files to record the specifications of play patterns (Pattern Library).
- **Online Detection:** After storing every pattern provided, the coach has to detect the pattern/s activated in the opponent team. The strategy of the opponent team was composed of several patterns which were given to the coach in the offline analysis. Hence, the coach should detect the play pattern/s of the opponent and report them. Also, the coach can advise its players in order to detect patterns better.

Hence, the performance of a given coach is based on its ability to detect and report patterns

(previously provided). The coaches can work both by analyzing logs of previous games and by observing and adapting while a game is proceeding. This process is shown in Figure 1.

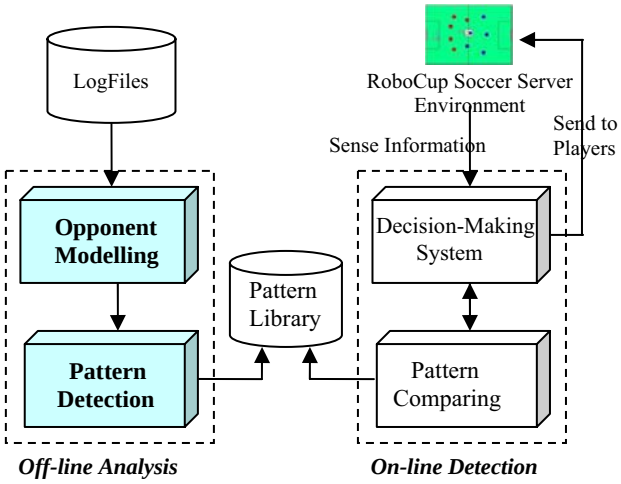

Figure 1. *Coach Competition* process.

3. OVERVIEW STRUCTURE OF OUR APPROACH

As this research is a preliminary work on this domain, the whole process have been divided and only the first phase of the process is considered (Offline analysis).

The inputs of our offline analysis are two games (*Pattern Log-File* and *No-Pattern Log-File*) played by a coachable team and an opponent team. However, a pattern has been activated in the opponent team of the game stored in the *Pattern Log-File*. Our work in this research is to check whether this pattern is related to the team formation. The figure 2 shows the overview structure of this work. Also, the two stages that divide this work are:

- Recognizing the formation of the opponent team from observations (in both games: *Pattern log-file* and *No-Pattern log-file*).

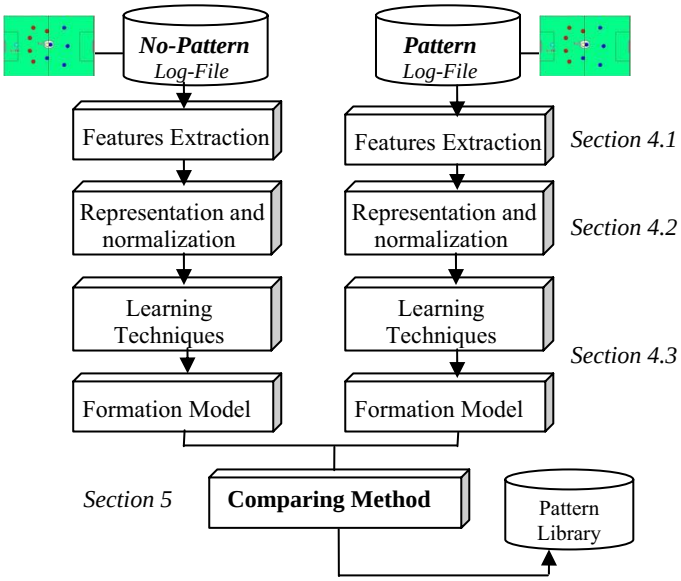

Figure 2. Overview structure.

- Comparing the two results to check whether they are not using the same formation. In this case, it means that the pattern activated in the opponent team (in the game *Pattern log-file*) is related to the team formation. Hence, the main formation characteristics of the *pattern log-file* must be stored in the Pattern Library, as a recognized pattern.

4. RECOGNIZING FORMATIONS

The recognition of team formations is an important concept in robotic soccer [Visser *et al*., 2000]. Druecker et al [Druecker *et al*., 2000] developed a neural network which is fed with the observed player positions and tries to classify them into a predefined set of formations. P. Riley [Riley, 2005] uses a coach that represents a formation as an axis aligned rectangle for each player on the team.

Because of a formation team is a very important part of a team strategy; a coach should be able to recognize the opponent team formation. Then, it should check whether the activated pattern in the opponent team is related to that formation.

The goal of this stage is to recognize a team formation based on observations. For resolving this objective, we use a classification algorithm based on supervised learning methods [Michalski, 1998; Mitchell, 1997]. In order to get examples that can be used in the learning method, we extract some data from the game logfiles. Then, two possible representations of the extracted features are proposed. These representations are normalized and finally, they are used to obtain a decision tree that classifies the different formations.

4.1 Features Extraction

In this first phase, the most relevant data of the log files must be extracted. For this purpose, we only extract information of the modelling team (opponent team). We get the following information:

- Cycle: A number that enables arrange the events.
- Opponent Positions: Each opponent's position is stored as X and Y axis coordinates.

Also, these data are extracted every cycle (a game consists of 6000 cycles). The next section describes how we use this data extraction.

4.2 Features Extraction Representation

In our approach, we have used two different representations of the features extractions:

Players Locations
The soccer field in *Online Coach Competition* is defined as a two dimensional coordinate system, so the location of a player is represented by two numbers that indicate the x and y coordinates of a point. The locations for each player on a team over one or more games are used in this first representation.

Hence, in this case, the input data of the learning method are represented by the position of each player in the field. The goalkeeper position is not considered because the formation of a team does not depend on this special player. According to this representation, the total amount of attributes considered by each example is 20 (10 players * 2 coordinates).

Players Locations Grid
The second features representation proposed is, as well, related to the general position of every player of a team in the field. However, this representation only considers the part of the field occupied by the every player of a team (except the goalkeeper). In this case, the features extracted are represented by a grid placed around every player position. This grid is divided into equal sized square cells. The number of cells of the grid is given to variation. In this work, we propose two grids of different amount of cells:

1. A grid of five by five cells which creates 25 cells of the same size.
2. A grid of ten by ten cells, as is shown in Figure 3.a.

In order to standardize this grid representation, we use the following considerations: The cells of the grid filled with at least one player are labeled as '1' and the empty cells of the grid are labeled as '0'. Hence, we obtain a vector with so many '1's and '0's as the cells of the grid. This vector has maximum ten '1's. The Figure 3.b shows the grid standardized corresponding to Figure 3.a.

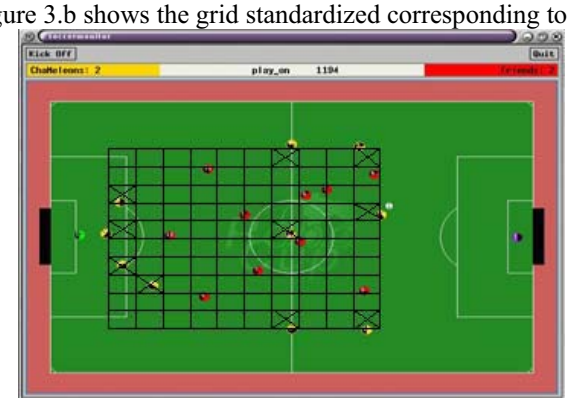

Figure 3.a. Grid of ten by ten cells around the player positions.

0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1

Figure 3.b. Standardized Grid Data.

4.3 Classification

In order to obtain the formation of a team and because of a learning technique is used; we need to consider what formations we could recognize. The different formations used in this research are taken from the Soccer Simulation Team "UvA Trilearn 2005" [Jelle et al., 2005]. The formations used in this research, are as follows:

- *0 = FT_UNKNOWN = 000*
- *1 = FT_INITIAL = 111*
- 2 = FT_433_OFFENSIVE
- 3 = FT_343_OFFENSIVE
- 4 = FT_DEFENSIVE = 442
- 5 = FT_OPEN_DEFENSIVE = 442
- 6 = FT_334_OFFENSIVE = 334
- 7 = FT_433_LINE_DEFENSE = 443
- 8 = FT_424_OFFENSIVE

Formation 0 and Formation 1 represent the unknown and the initial formations, so they have not been considered in this research. The other seven formations are noted as three numbers: (1) defense players (2) mid-field players and (3) forward players.

For recognizing the seven possible formation of a team, we can use a classification method. The method used in this approach is the algorithm C4.5 [Quinlan, 1993]. This algorithm is a standard benchmark in machine learning.

To get the input files of the algorithm C4.5, we used the logfiles of games in which a team follows during the whole game a specific formation. Using the features extraction process, we obtain multiple records. Each record has the same structure, consisting of a number of attribute/value pairs. One of these attributes represents the category of the record. In this case, the category of the record is the formation of the team.

In this research, as we have seen in the above section, two different set of features are used as input data of the algorithm C4.5. The first set of features (Player Locations) describes the location for each player and takes a numeric value. The second set of features (Player Location Grid) is a grid of different amounts of cells and takes a vector of values {0, 1}.

In order to extract these features from the log files, we have fully implemented a program that is able to take snapshot every cycle and extracts the positions of the ten players (the goalkeeper is not included) of a

team. The result is a record with 20 real numbers (x-y position of the players) per cycle. Also, if it is necessary, this program normalizes these data. As every game is played according to a specific formation defined by us, we can label every record obtained from the snapshot with the defined formation. Because of a snapshot is made once per second, we can get a large quantity of records.

After extracting and normalizing features, we apply the C4.5 algorithm using the tool WEKA version 3.4 [Witten and Frank, 2000]. This tool implements several data mining and machine learning algorithms, including classification algorithms as C4.5.

4.4 Results

The following experiments are based on 2100 records (300 records for each formation) obtained from the logfiles of seven games (each game is played with a different formation).

As we consider 3 different features representation (Players Location, 5x5 Grid and 10x10 Grid), we create 3 different datasets. For each dataset, a pairs of training and testing set is created with 10-fold cross-validation. Then, C4.5 algorithm is run.

Table 1 shows the accuracy for C4.5 classifier with the three different sets of input data.

Table 1. Accuracy for C4.5 classifier

Input Data Set / C4.5Results	Players Location	Players Locations Grid 5x5 Grid	
		5x5 Grid	10x10 Grid
Number of features	20	25	100
Correctly Classified Instances	98.462 %	89.139 %	93.671 %
Number of Leaves	206	543	922
Size of the tree	411	1085	1843

Analyzing the results, we observe that, if the input data of the classifier is the location of each player (*x* and *y* coordinates), then the correctly instances can be classified very accurately. Also, in this case, the number of leaves of the obtained tree is not big (206), so it can be easily converted into rules. But, if we use players locations to classify the formation and the players change their role, the result could not be viable. For example, if the player 5 is considered as a defender but its role is changed to forward, the rules obtained from this algorithm could be inexact. Hence, the use of this features representation depends on if the role of a player could be changed.

On the other hand, if we use the Players Location Grid, the role of the players is not considered, so the result obtained from the C4.5 classifier is always viable. Although the result of correctly classified instances is better using a 10x10 grid, the size of the decision tree is bigger than using a 5x5 grid. Hence, we can use different features representations as input data set depending of the purpose of the work.

5. COMPARING METHOD FOR PATTERN RECOGNITION

The second stage of this research is to propose a comparing method of two agent models. Although this method can be used in several environments, the agent models in this work are two soccer games.

In this method, the game in which the opponent team uses a general strategy will be represented by *A*, and the game in which a pattern has been activated in the opponent team, will be represented by *B*. Hence, the goal of this method is to check if the opponent team has or not the same formation in the two games (*A* and *B*). If the formation is different, a pattern has been recognized.

Also, a game is represented by set of variables (formation) per cycle. And so, we can contrast whether two games are following the same distribution. In order to get this comparison, we use statistical analysis.

As we must recognize the opponent team formation during the whole game, this formation is obtained every cycle and stored in a vector. The size of this vector is the same as the number of cycles of a game. We define a 1 x *n* vector (*n*=number of cycles) to represent the opponent team formation every cycle. The notation used in this case, is: *FC_Game,* where *FC* indicates the opponent team formation in the cycle *C*, and *Game* indicates the considered game (*A* or *B*).

As in our case, the number of cycles is 6000; the representation of the vector for the game A is:

OppTeamGameA = {*F1_A, F2_A, … , F6000_A* }

And the vector for the other game B, is as follows:

OppTeamGameB = {*F1_B, F2_B, … , F6000_B*}

In order to compare the obtained data, we use a statistical test. As we need to compare two groups of sampled data (*OppTeamGameA* and *OppTeamGameB*), a non-parametric test is used. In our research, the most appropriate test is the *rank-sum* test [Bradley, 1968] (also called the Mann-Whitney test). Unlike parametric *t-test* [Iman and Conover, 1983], the *rank-sum test* makes no assumptions about the distribution of the data. The distribution of the data is not relevant in our case because the order of the formations during the game is not relevant. The *rank-sum* test, like many non-parametric tests, uses the ranks of the data rather than their raw values to calculate the statistic.

In this method, the hypotheses for the comparison of our two groups are:

- *H0*: The opponent team formation of the two games (*A* and *B*) is the same.
- *H1*: The opponent team formation of the two games (*A* and *B*) is not the same.

In order to apply this test, the data vectors (*OppTeamGameA* and *OppTeamGameB*) must first be combined into a set of *n* elements. In our case *n* = 12000 (6000+6000). Then, these elements are ranked from lowest to highest (in our case, if the formations are represented by a name, the elements can be ranked lexicographically). Finally, after applying this test fully, we obtain the value *U*. This value is compared to a table of critical values for *U* based on the sample size of each group. If *U* exceeds the critical value for *U* at some significance level (usually 0.05), it means that there is evidence to reject the null hypothesis in favour of the alternative hypothesis.

Although it is not considered in this research, if we would want to compare the formation in more than a game, there would be more than two groups in this comparison, and the test becomes a Kruskal-Wallis test.

6. CONCLUSIONS

A novel method to compare two different agent behaviours in a multi-agent system has been presented. The environment where our method has been used is *RoboCup Online Coach Competition.*

In this environment, we have divided our research into two parts: In the first part, a classification algorithm to recognize formations in the soccer domain is developed in depth. In the second part, a comparing method of two agent models based on formations is presented.

The comparing method proposed in this research, is a preliminary work to tackle the goal of the *RoboCup Online Coach Competition*. The novelty of this goal, lead us to divide the goal and consider only the first part.

The development and testing of this work, has demonstrated how classification algorithm and statistical test can be successfully integrated for formation recognition in the multi-agent system of the *RoboCup Online Coach Competition* domain. Although the comparative method explained in this research is used in the RoboCup environment, it could be used in many others.

REFERENCES

[Bradley, 1968] Bradley J. V., *Distribution-Free Statistical Test*, Prentice-Hall: Englewood Cliffs, N.J. 1968.

[Carmel and Markovitch, 1996] D. Carmel & S. Markovitch. Opponent modelling in a multi-agent system. *In G. Weiss and S. Sen, editors, Lecture note in AI, 1042: Adaptation and Learning in Multi-agent Systems, Lecutre Notes in Artifiical Intelligence*. Springer-Verlag, 1996.

[Chen *et al.*, 2001] M. Chen, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang & X. Yin. Soccerserver Manual v7. *The RoboCup Federation*, 2001.

[Druecker *et al*., 2000] C. Druecker, C. Duddeck, S. Huebner, H. Neumann, E. Schmidt, U. Visser, & H.-G. Weland. Virtualweder: Using the online-coach to change team formations. *Technical report, TZI-Center for Computing Technologies*, University of Bremen, 2000.

[Iman and Conover, 1983] Iman, R. L. & J. Conover, *A Modern Approach to Statistics*, John Wiley and Sons: New York, 1983.

[Jelle *et al.*, 2005] Jelle R. Kok and Nikos Vlassis. UvA Trilearn 2005 Team Description. In *Proceedings CD RoboCup 2005,* Springer-Verlag, Osaka, Japan, July 2005.

[Kitano *et al*., 1997a] H.Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda & M. Asada. The robocup synthetic agent challenge. *In Proceedings of the Fifteenth Internacional Joint Conference on artificial Intelligence (IJCAI97)*, pages 24-49, San Francisco, CA, 1997.

[Kitano *et al.*, 1997b] H, Kitano, M. Asada, Y. Kuniyoshi, I. Noda & E. Osawa. Robocup: The robot world cup initiative. *In W. Lewis Johnson and Barbara Hayes-Roth, editors, Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340-347, New York, 5-8, 1997. ACM Press.

[Kuhlmann *et al.*, 2004] G. Kuhlmann, P. Stone & J. Lallinger. The Champion UT Austin Villa 2003 Simulator Online Coach Team. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. *RoboCup-2003: Robot Soccer World Cup VII*, Springer Verlag, Berlin, 2004.

[*Ledezma et al.*, 2005] A. Ledezma, R. Aler, A. Sanchis & D. Borrajo. *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*, chapter Predicting Opponent Actions by Observation, pages 286-296. Springer Verlag, Lisbon (Portugal), 2005.

[*Riley et al.*, 2002a] P. Riley, M. Veloso & G. Kaminka. Towards any-team coaching in adversarial domains. In Onn Shehory, Thomas R. Ioerger, Julita Vassileva, and John Yen, editors, P*roceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Sytemsi (AAMAS)*, Bologna, Italy, 2002.

[Michalski, 1998] Michalski R.S, *Machine Learning and Data Mining, Methods and Applications*. Wiley, 1998

[Mitchell, 1997] Mitchell T.M., *Machine Learning*, McGraw-Hill, 1997

[Quinlan, 1993] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Riley and Veloso, 2000] P. Riley & M. Veloso, *Distributed Autonomous Robotic Systems,* volume 4, chapter *On Behavior Classification in Adversarial Environments*, pages 371-380. Springer-Verlag, 2000.

[*Riley et al.*, 2002b] P. Riley, M. Veloso & G. Kaminka. An Empirical Study of Coaching. In H. Asama, T. Arai, T.Fukada, and T. Hasegawa, editors. *Distributed Autonomous Robotic Systems 5,* pp 215-224, Springer-Verlag, 2002.

[Riley, 2005] P. Riley. Coaching: Learning and Using. *Environment and Agent Models for Advice*. January 11, 2005.

[RoboCup Organizing Committee, 2005] The RoboCup 2005 Simulation League Organizing Committee. *RoboCup 2005 Official Rules for the Coach Competition*. January, 2005.

http://staff.science.uva.nl/~jellekok/robocup/rc05/Coach-Rules-Revision0.3.pdf

[Scerri *et al.*, 2001] P. Scerri, D. Pynadath, & M. Tambe. Adjustable autonomy in real-world multi-agent environments. *In Agents-2001*, 2001.

[Steffens, 2002] T. Steffens. Feature-based declarative opponent-modelling in multi-agent systems. *Master´s thesis, Institute of Cognitive Science Osnabrück*, 2002.

[Stone and Veloso, 2000] P. Stone, P. Riley, and M. Veloso. Defining and using ideal teammate and opponent agent models. *In Proceedings of the Twelfth Innovative Applications of Artificial Intelligence Conference (IAAI-2000)*, 2000.

[Turocy and Von Stengel, 2001] T. L. Turocy & B. Von Stengel. Game Theory. *CDAM Research Report LSE-CDAM*., 2001

[Visser *et al.*, 2000] Visser U., Drücker C., Hübner S., Schmidt E. and Weland. Recognizing Formations in Opponent Teams. *Proceedings of the RoboCup 2000, Robot Soccer World Cup IV*, Melbourne, Australia, pp. 391-396.

[Witten and Frank, 2000] I. H. Witten, *E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, San Francisco, 2000.