The Winning Advantage: Using Opponent Models in Robot Soccer

José Antonio Iglesias, Juan Antonio Fernández, Ignacio Ramon Villena, Agapito Ledezma, and Araceli Sanchis

Carlos III University of Madrid,
Avda. de la Universidad, 30, 28911 Leganés (Madrid), Spain
jiglesia@inf.uc3m.es, {100048324,100048430}@alumnos.uc3m.es,
{ledezma,masm}@inf.uc3m.es

Abstract. Opponent modeling is a skill in multi-agent systems (MAS) which attempts to create a model of the behavior of the opponent. This model can be used to predict the future actions of the opponent and generate appropriate strategies to play against it. Several researches present different methods to create an opponent model in the RoboCup environment. However, how these models can impact the performance of teams is an essential aspect. This paper introduces a novel approach to use efficiently opponent models in order to improve our own team behavior. The basis of this approach is the research done by *CAOS Coach Team* for modeling and recognizing behaviors evaluated in the *RoboCup Coach Competition 2006*. For using these models, it is necessary a special agent (*coach*) which can model the observed opponent team (based on the previous research) and communicate a counter-strategy to the coached players (using the approach proposed in this paper). The evaluation of this approach is a hard problem, but we have conducted several experiments that can help us to know if we are going in a promising direction.

1 Introduction

Humans usually try to predict the behavior of others to interact with them efficiently. This prediction is also very interesting in the multi-agents systems (MAS) with adversary agents, in which the agents need to interact and cooperate with other agents in order to increase its ability to compete. Opponent modeling is a skill in MAS which attempts to create an opponent behavior model. Nowadays, opponent modeling is increasingly becoming more significant and complex.

Robot World Cup (RoboCup) was proposed in 1994 and one of its challenges was the opponent modeling: *The RoboCup opponent modeling challenge calls for research on modeling a team of opponents in a dynamic, multi-agent domain* [1]. Because of the dynamical and adversarial nature of a soccer play, opponent modeling is very relevant in the RoboCup environment, especially in the simulation league. In 2001, a new competition was created: *RoboCup Coach Competition*, in which an online *coach* was able to act as an advice-giving agent [2].

This paper proposes a novel technique for using opponent modeling in a coachable team (team whose players can be advised by the coach) of the RoboCup

environment in order to improve its performance. This technique is based on the opponent modeling approach proposed in [3]. Therefore, our main contribution is to describe how to create a counter-strategy (to improve the behavior of our simulated soccer team) using a model of the opponent team.

The rest of the paper is organized as follows. In section 2 we provide a brief overview of the related work on opponent modeling in the RoboCup Simulation. The opponent modeling approach used in the paper is summarized in section 3. The presented technique to generate counter-strategies is presented in section 4. Experimental results are given in section 5. Finally, section 6 contains future work and concluding remarks.

2 Opponent Modeling in the RoboCup Soccer Simulation

RoboCup Soccer Simulation provides a good platform for modeling a soccer team in a dynamic and multi-agent domain. In the 2D Soccer Simulation, Kaminka et al. [4] present a hybrid approach to learn the coordinated sequential behavior of teams, from a time-series of continuous observations. Wünstel et al. [5] propose an approach in which each agent observes and recognizes the behavior of the adversaries. However, a frequently used opponent modeling approach in the RoboCup environment is to rely on a privileged agent (*Coach*) which can advice to the coached teammates (*coachable team*) about how to act to improve its performance. This communication is accomplished by a standard coach language called *CLang* [6]. A *coah* has a full noise-free view of the field (it gets global and noiseless information from the Soccer Server about the position and speed of all players and the ball) and one of its main advantages is that it has access to logfiles of past games played by the team to model (*opponent team*).

In order to focus entirely on opponent modeling, the *RoboCup Simulation Coach Competition* was held from 2001 to 2006. This competition is situated within the same soccer server, but instead of creating a full soccer team, a single *coach* agent must be implemented. Several works [7,8] present coaching techniques for a simulated robotic soccer domain and justified that coaching can help teams improve in this domain.

RoboCup Coach Competition changed in 2005 in order to emphasize opponent-modeling approaches. In this competition, the teams are directly evaluated based on how its *coach* agents identify the weaknesses and strengths (*patterns*) of the opponent from other opponent behaviors without these patterns (*base strategy*). After detecting the different *patterns* of different teams, the *coach* is rated on how well it recognizes them by observation. Using this environment, Kuhlmann et al. [9] model a soccer team by characterizing their behavior with a set of features calculated from statistics gathered while observing a game. The winners of the RoboCup Coach 2006 Competition present in [10] a learning architecture for modeling the opponent and a rule based expert system architecture to provide a provoking strategy for opponent players. Recently, Iglesias et al. [3] present a novel method used by the *CAOS team* to model and recognize successfully the behavior of a soccer team.

3 The Opponent Modeling Approach

This paper is based on the approach presented by Iglesias et al. [3,11] for modeling and recognizing the opponent behavior in the *RoboCup Coach Soccer* environment (used by the *CAOS Coach Team*). This section outlines this method and the environment in which it was evaluated: *RoboCup 2006 Coach Competition*. According to the official rules [12] of this competition, previously to the competition, a set of base strategies are created and some matches in which the opponent team follows this strategy are played (*no-pattern log-files*). Then, some patterns are added to these base strategies of the opponents, and some sample matches are played again (*pattern log-files*). Many pairs of log-files (*pattern log file* and its corresponding *no-pattern log file*) are created.

Each *coach* team participant is provided with several *pattern log-files* (only one pattern is activated in a log-file) and its corresponding *no-pattern game log-files*. This competition goal is to look for the qualitative differences among the pattern and its corresponding no-pattern log file to identify the weaknesses or strengths defined in the pattern. Once every pattern has been detected and stored, the *coach* should recognized them by observing a live game (in which the opponent follows 3 or 4 patterns). This competition consists of two phases (*Off-Line Analysis* and *On-Line Recognition*) which are tackled in [3] as follows:

Off-Line Analysis: The observation stream of the whole game is transformed into an ordered sequence of recognized atomic behaviors. Based on a work of Kuhlmann et al. [13], the eight atomic soccer actions inferred are: *Pass*, *Dribble*, *Intercept Pass*, *Steal*, *Goal*, *Missed shot*, *Foul* and *Hold*. This sequence of soccer actions is segmented into several subsequences of the same length. Using these subsequences, a *trie* representation is done (the usefulness of this structure is demonstrated in [4]). Then, the relevance of each subsequence of soccer actions is calculated using its relative frequency or support. Thus, a team behavior model is represented as a set of subsequences labeled with their support where each sequence represents an action and the actions previously executed. These subsequences and its support can be created using a visual tool that we have developed called *Viena*[1]. In addition, as the pattern is represented by the difference between two team behavior models, a method for this comparison is proposed in the approach. The result of this comparison is a set of relevant subsequences (*pattern*) and it is stored in the *CAOS Pattern Library*.

On-Line Recognition: A live game, in which the opponent team follows a few of the pattern previously analyzed, is observed by the coach and it must recognized and reported these patterns. Firstly, the observations of the opponent team are collected and its behavior model is created as explained above. Then, this model is matched with all the patterns stored in *CAOS Pattern Library*. By these comparisons, the patterns with a closer similarity to the observed game are recognized as the activated patterns in the opponent team.

Results: The results of this competition and how it is calculated the performance of a given *coach* is detailed in [3].

[1] Available at: http://www.caos.inf.uc3m.es/~viena/

4 Choosing a Counter-Strategy

After detecting a pattern executed by the opponent team, the coachable team will be advised by the *coach* to execute a counter-strategy in order to improve its performance. According to the previous section, a pattern consists of several soccer actions executed by the players and its corresponding prefix of actions (actions previously executed). In this section, we explain how we can generate a counter-strategy when we have identified the opponent behavior into a specific pattern. In this case, the counter-strategy (that is executed by the coachable team) consists of several counter-actions as response to the recognized actions (of the opponent teams). Based on how the recognized action can be counter-acted in order to obtain improve our strategy, we have divided the recognized actions in two classes: those in which the player who execute the action is important (*dribble, hold, shoot, goal*) and those which the next ball possessor is very important(*pass*). The proposed counter-actions (according to the soccer actions detected in [3]) for these two kinds of actions are as follows:

- ***Recognized action:*** Pass
  Counter-Action to execute: To anticipate the pass to intercept the ball before it reaches the receiver.
- ***Recognized action:*** Dribble, Hold, Shoot, Goal
  Counter-action to execute: To detect the player who is executing the action and intercept the ball.

Once we know the action that an opponent player will probably execute, the coachable player who will execute the corresponding counter-action must be chosen. For this task, each coachable player is labeled with its corresponding role (defender, midfielder, striker or goalkeeper) and its zone in the field (upper or lower). Then, the coachable player who will execute the action should be in the same field zone of the opponent player and its role should be a counter-role of the opponent player. The counter-role of a defender is a striker and vice versa. In addition, the selected coachable player should have not been previously chosen to execute another counter-action in the same conditions.

Finally, *CLang* language is used to inform and advice to the coachable players what they should do. Thus, the counter-action is sent to the corresponding player by the *coach* using *CLang*. This language is suited to represent strategies because its messages are basically production rules mapping conditions to actions: *CLang* conditions are constructed from logical connectives (and, or, not) of descriptions of the world state like player and ball positions, play modes, scores, and time. *CLang* actions are designed to have relatively clear semantics and are recommended macro-actions for the players such as *position-in-regions*, *marking*, *passing-to-regions*, *passing-to-players*, *dribbling*, *intercepting* and *tackling*. As an example, the following lines described a rule in *CLang*:

```
1.  (definer REGION_A (rec (pt 30 20)(pt 40 35)))
2.  (definerule RuleNumber1 direc
3.  (and (bowner opp 2) (playm play_on) (bpos REGION_A))
4.  (do our 2 (markl7)))
```

Table 1. CLang Rules from the Observed action and its prefix

#	Action	Prefix	CLang Rule
1	**Pass_Opp06_Opp07**	None	*definerule PREVENT_PASS_06toO7 direc((bowner opp{6}) (do our{5} (mark {7})))*
	Observations: When the opponent 6 is the ball owner, our coachable player 5 marks opponent player 7.		
2	**DHSG_Opp08**	None	*definerule PREVENT_DHSG_O8 direc((bowner opp{8}) (do our{3} (intercept)))*
	Observations: When the opponent 8 is the ball owner, our coachable player 5 intercepts (tries to get the ball).		
3	**Pass_Opp03_Opp04**	Pass_Opp02_Opp03	*definerule PREVENT_PASS_O4 direc((bowner opp{2 3}) (do our{2} (mark {4})))*
	Observations: When one of the opponents 2 or 3 is the ball owner, our coachable player 2 marks opponent player 4.		
4	**DHSG_Opp03**	Pass_Opp02_Opp03	*definerule PREVENT_DHSG_O3 direc((bowner opp{3}) (do our{10} (intercept)))*
	Observations: When the opponent 3 is the ball owner, our coachable player 10 tries to get the ball.		
5	**Pass_Opp05_Opp08**	DHSG_Opp05	***1.*** *definerule PREVENT_DHSG_O5 direc((bowner opp{5}) (do our{4} (intercept)))* ***2.*** *definerule PREVENT_Pass_O5 direc((bowner opp{5}) (do our{5} (mark {8})))*
	Observations: Two CLang rules are created: When opponent 5 is the ball owner, our coachable player 4 intercepts and our player 5 marks opponent player 8		
6	**DHSG[1]_Opp010**	DHSG[2]_Opp10	***1.*** *definerule PREVENT_DHSG[1]_10 direc((bowner opp{10}) (do our{2} (intercept)))* ***2.*** *definerule PREVENT_DHSG[2]_10 direc((bowner opp{10}) (do our{3} (intercept)))*
	Observations: The actions represented by DHSG[1] and DHSG[2] must be different. A CLang rule is created for each action executed by opponent 10.		

The first line defines a region (a rectangle area) in the field. In the next 3 lines, the rule named *RuleNumber1* is defined: The second line is the beginning of the rule and it is due to the *coach* protocol. In the third line the situation description is detailed and denotes that: the opponent player 2 has the ball, the play mode is *on*, and the ball is in a region defined in the first line. In the last line, the action is specified: the player 2 marks the opponent player 7 (marking is a standard soccer term meaning to play defense against a player).

In order to explain how the counter-action is generated, the table 1 detailed some examples in which the *CLang* rules activated are calculated from the detected action and its prefix of actions. In this case, the coachable players who will execute the counter-action have been chosen arbitrarily. Each action is labeled with the number of the player or players who played the action (for example, the action *PA_Opp06_Opp08* indicates that the opponent player 6 passes to the opponent player 8). The recognized actions are: *Pass, Dribble*, *Hold, Shoot* and *Goal*. However, the response for the actions: *Dribble*, *Hold*, *Shoot* and *Goal* is the same. Thus, any of these actions is indicated in the table as the action DHSG.

5 Experimental Setups and Results

In order to evaluate the technique proposed in this research, we conducted extensive experiments using two different environments.

5.1 Data From 2006 Coach Competition

The first experiments were conducted using the patterns and the platform proposed in the *RoboCup 2006 Coach Competition*[2]. This competition consists of 3 rounds and each round consists of 3 iterations. In this section, our results in the first iteration of the first round are explained. Firstly, it is necessary to analyze all the patterns that can be later activated in the on-line game. In this round, 17 different patterns are analyzed by our *coach* in the *off-line analysis* (as explained in section 3). Then, the first iteration of this first round is executed. In this iteration, 4 different patterns (of the 17 analyzed patterns) are activated in the opponent team during a game (6000 cycles). Thus, our coachable team should recognize these 4 patterns and then, send to the coachable players the appropriate counter-actions.

Round 1. Iteration 1. In this iteration, our *coach* team recognizes 2 patterns (*pattern15* and *pattern16*) which are describedin *CLang* in table 2.

Table 2. Patterns recognized by the coach - CLang Rules

CLang Rules: **Pattern15**	CLang Rules: **Pattern16**
definerule RULEPASS12 direc((bowner our{1})(do our{1} (pass{2})))	definerule RULEPASS12 direc ((bowner our{1})(do our{1} (pass {3 4})))
definerule RULEPASS210 direc(((bowner our{2})(do our{2} (pass{10}))))	definerule RULEDRIBBLE3 direc ((bowner our{3}) (do our{3} (dribble (pt -52 33))))
definerule RULEDRIBBLE10 direc ((and (playm play_on)(bowner our{10})) (do our{10} (pass ((pt ball)+(pt 0 20)))))	definerule RULEDRIBBLE4 direc ((bowner our{4}) (do our{4} (dribble (pt -52 -33))))

Figure 1 shows the counter-actions sent by our *coach*. In this case, the counter-actions proposed for the *pattern15* are correctly created and sent to the coachable players in the cycle 1500. The counter action of the first rule of the *Pattern15* (*opponent player 1 passes to opponent player 2*) is: *when the opponent 1 is the ball owner, the coachable player 9 should mark to the opponent player 2*. The other 2 counter-actions activated causes that *the coachable player 3 marks to opponent 10 when the opponent 2 is the ball owner*, and that *when the opponent player 1 is the ball owner, the coachable player 8 should intercept*.

[2] All the files used in this competitions are available at the RoboCup Game Logs Page: http://sserver.jpn.org/RoboCup/log/RoboCup06/Coach/

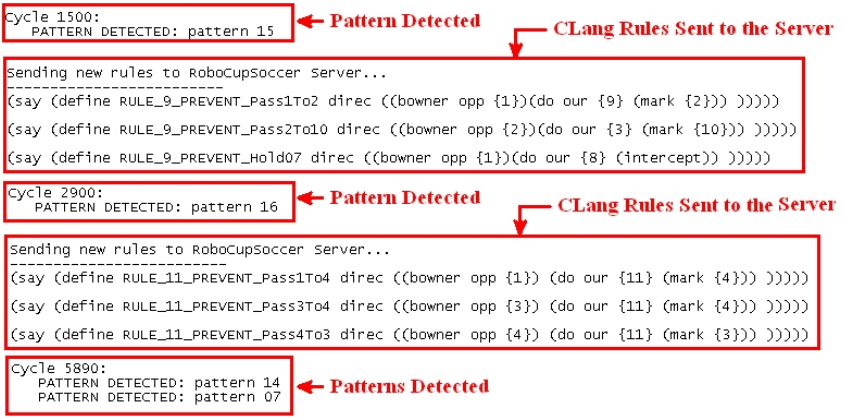

Fig. 1. Our results in the first iteration of the RoboCup Coach Competition 2006

However, the *pattern16* is detected in the cycle 2900 (figure 1), but it was not activated in the opponent team (this pattern recognition is wrong). The counter-actions generated for the detection of the pattern16 are correct but the counter-strategy is not completely valid for that opponent.

In the last cycles of the game (cycle 5900), other two patterns are recognized (*pattern 14* and *pattern17*). In this case, *pattern14* is correctly recognized but *pattern17* is wrongly recognized. Since the recognition of these patterns is done very late, the counter-strategies are not sent to the coachable players.

Observations: Although in this paper all the results of the *2006 Coach competition* are not shown because of lack of space, after analyzing the different rounds in this competition, we can remark observing these preliminary results the importance of the pattern recognition: The counter-actions of a pattern can be very useful but if the pattern is not recognized, those counter-actions are never used. However, when a pattern is detected is because the opponent team behaves *similar* to the pattern; therefore, the counter-actions generated can be useful for improving the strategy of our coachable team although the pattern has not been correctly detected.

5.2 Data from a Team General Behavior

In these experiments, instead of creating the counter-actions for a specific pattern in the opponent team, we propose the creation of counter-actions for a general behavior of a team (what we call general strategy).

Observations: Regarding *CLang* language, we realized that if we use rules in which the condition that a player is owner of the ball (*bowner*) is used, the corresponding action of the rule is usually activated very late (when the coachable

player knows that he should do a counter-action, the opponent ball owner could have changed and the counter-action could be not useful). For this reason, the *bowner* condition was changed in order to anticipate to the ball owner before he really is. We consider that when the ball is *near* a player (using a circle with center point the ball position and a radius of 12 meters), it could be the next ball owner and the counter-action should be executed. Therefore, the condition: *(bowner our {X})* was changed by: (ppos our {X} 1 11 (arc (pt ball) 0 12 0 360)). In addition, the *mark* action was changed by the *markLine* action (stay between the ball and one player) because of its effectiveness.

6 Conclusions and Future Work

Adaptation and learning abilities are essential for an intelligent agent that interacts with other selfish agents. We presented a technique used in the RoboCup Soccer Simulation environment to generate counter-actions (that can be executed by our soccer team) when we know how the opponent team behaves. For this technique it is necessary a special agent (*coach*) which can model the opponent team and create an on-line counter-strategy according to this model. *CLang* language is used to advise to the players the actions they should executed.

This technique is based on the opponent modeling approach used in the *RoboCup 2006 Coach Competition* by the *CAOS Coach Team* and the main goal is to use efficiently the opponent models created by that *Coach*. This technique is a necessary continuation of the goal proposed in that competition. The evaluation of this technique (to know if the impact of a counter-strategy is effective in the performance of a team), is a hard problem. Nevertheless, the preliminary experiments shown in this paper seems to be going in a promising direction.

Acknowledgments. This work has been supported by the Spanish Government under project TRA2007-67374-C02-02.

References

1. Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., Asada, M.: The robocup synthetic agent challenge 97. In: IJCAI 1997 (1997)
2. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer server: a tool for research on multiagent systems. In: Applied AI, vol. 12, pp. 233–258. Taylor and Francis, Abington (1998)
3. Iglesias, J.A., Ledezma, A., Sanchis, A.: Caos coach 2006 simulation team: An opponent modelling approach. Computing and Informatics 28(1), 57–80 (2009)
4. Kaminka, G.A., Fidanboylu, M., Chang, A., Veloso, M.M.: Learning the sequential coordinated behavior of teams from observations. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 111–125. Springer, Heidelberg (2003)
5. Wünstel, M., Polani, D., Uthmann, T., Perl, J.: Behavior classification with self-organizing maps. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 108–118. Springer, Heidelberg (2001)

6. Chen, M., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., Yin, X.: Soccerserver manual ver. 7 (2002)
7. Riley, P., Veloso, M., Kaminka, G.: An empirical study of coaching (2002)
8. Steffens, T.: Feature-based declarative opponent-modelling in multi-agent systems, Master's thesis, Institute of Cognitive Science Osnabrück (2002)
9. Kuhlmann, G., Knox, W.B., Stone, P.: Know thine enemy: A champion robocup coach agent. In: Proc. 21st National Conference on AI, pp. 1463–1468 (2006)
10. Fathzadeh, R., Mokhtari, V., Kangavari, M.R.: Opponent provocation and behavior classification: A machine learning approach. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 540–547. Springer, Heidelberg (2008)
11. Iglesias, J.A., Ledezma, A., Sanchis, A., Kaminka, G.A.: Classifying efficiently the behavior of a soccer team. In: Burgard, W., et al. (eds.) IAS-10, pp. 316–323 (2008)
12. Committee, R.: Robocup 2006 official rules for the competition (December 2006), `http://www.caos.inf.uc3m.es/caoscoachteam/otros/rules0.0.pdf`
13. Kuhlmann, G., Stone, P., Lallinger, J.: The champion UT Austin Villa 2003 simulator online coach team. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020. Springer, Heidelberg (2004)