A Comparing Method of Two Team Behaviours in the Simulation Coach Competition

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. The main goal of agent modelling is to extract and represent the knowledge about the behaviour of other agents. Nowadays, modelling an agent in multi-agent systems is increasingly becoming more complex and significant. Also, robotic soccer domain is an interesting environment where agent modelling can be used. In this paper, we present an approach to classify and compare the behaviour of a multi-agent system using a coach in the soccer simulation domain of the RoboCup.

1 Introduction

Modelling agent techniques have been used from Artificial Intelligence (IA) beginnings. The idea of extract knowledge from other agents' behaviour was used, originally, in the field of the game theory [1]. After not considering information about opponents in game mechanics, it was realized that knowledge about the opponent's strategy can enhance the game results. After this first research, many others researches have been working to develop different ways of modelling other agents. Carmel and Markovitch [2] 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) [3] 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 [4]. One of the challenges for the near future is opponent modelling, a research which can be used both RoboCup and general multi-agent system.

One of the leagues competitions in RoboCup is *Simulation league*. In this league there are three competitions; *2D, Coach* and *3D* and each of them has different goals.

In the *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. [5] 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. [6] present an approach to modelling low-level behaviour of individual opponent agents, OMBO (Opponent Modelling Based on Observation). Druecker et al [7] develop a neural network which is fed with the observed player positions and tries to classify them into a predefined set of formations. Riley et al. [8] propose a classification of the current adversary into predefined adversary classes. Based on this work, Steffens [9] presents a feature-based declarative opponent-modelling (FBDOM) which identify tactical moves of the opponent.

The *RoboCup Coach Competition* mainly deals with an agent (coach) providing advices to another agents about how to act. About the Coaching problem raised in *Coach Competition*, Riley et al. [11] present a general description of the coaching problem. This description is the first step in understanding advice-based relationships between automated agents. In this research, one of the reasons 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 [12]. Also, Riley et al. [13] justify that coaching can help teams to improve in simulated robotic soccer domain.

RoboCup Coach Competition changed recently in order to emphasize opponent-modelling approaches. The main goal of this competition is to compare two team behaviours, but in one of them a play pattern[1] (way of playing soccer) has been activated but not in the other one. With regards to this competition, our work on opponent modelling is driven by the goal of learning the behaviour of agents by observing them. Also, this behaviour must be stored in a way that it can be compared to another one.

This paper introduces a comparing process of agent behaviours as a general framework which can be used in different multi-agent systems. Our proposal is implemented and empirically evaluated in the RoboCup domain, specifically in the *Coach RoboCup Competition*.

The goal of this research has several aspects:

- Abstracting useful features from the multi-agent system log files.
- Analyzing these features in order to recognize different events.
- Proposing the storage of the events in a trie data structure.
- Comparing different tries to get the useful information.
- Recording this information in an easy way.

This paper is organized as follows: Section 2 presents a summary on our approach to agent modelling. Section 3 provides the trie data structure used to store events. Section 4 describes the proposed comparing method. The experimental results are shown in section 5. Finally, conclusions and future works are drawn in section 6.

[1] In this paper we use the term pattern as a contraction of play pattern.

2 The Modelling Environment

2.1 The RoboCup Simulation League

The robotic soccer domain has been chosen because is one of the most used and useful domains in this subject. The *RoboCup Simulation League* uses the Soccer Server System [10] 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 goalie) 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.

2.2 The RoboCup Online Coach Competition

The *RoboCup Online Coach Competition* is a sub-league for automated coaches which are able to advise a team of autonomous agents to perform better its behaviour in front of an opponent. In this case, in addition to the players, each team may connect a privileged client to the server, the coach agent. It gets global and noiseless information from the Soccer Server about the position and speed of all players and the ball. This coach can only support its team by giving messages to its player in a standard coach language called *CLang*, which was developed by members of the simulated soccer community [10].

This competition structure changed recently in order to emphasize opponent-modelling approaches. According to the RoboCup 2005 official rules for the *Coach Competition* [20], instead of having a coachable team playing against an opponent to score, this coachable team plays against a fixed opponent in which several playing patterns have been activated. The term pattern is used to describe a simple behaviour that a team performs which is predictable and exploitable for the coaches [14]. For example, a possible pattern could be that the player number 2 marks the attacker advancing from the lower part of the field. The coach is given a number of game logs and it must model them in order to detect the used patterns and report them. In other words, the coaches should be looking for the qualitative differences among the pattern log file and the corresponding no-pattern log file to recognize the pattern correctly.

Therefore, this competition requires two phases:

- **Offline analysis:** Is the first phase of each round and the coaches analyzes the log files of the patterns which will be used during the round.
- **Online Detection:** The only task of coach in this phase is to detect the pattern(s) activated in the opponent team.

The performance of a given coach is based only on its ability to detect and report patterns. The research focus is on team/opponent modelling and online adaptation. The coaches can work both by analyzing logs of previous games and by observing and adapting while a game is proceeding.

3 Classifying Behaviour

As we describe in the above section, *RoboCup Online Coach Competition* has two phases, but in this paper we consider only the first one (offline analysis) because this is a preliminary work on this domain. The goal of this first process is to extract and store in a useful way the important data from log files. This data must be relevant to classify the multi-agent/team behaviour.

3.1 Feature Extraction

Kuhlmann et al. [14] describe a procedure to identify high-level events in a play game. An event represents a recognized atomic behaviour. The goal of their work is to get a coach that learns to predict agent behaviour from past observations and generates advices to improve its team's performance. Based on that work, we carried out two processes: feature extraction and event recognition. Unlike Kuhlmann procedure, we use the result of these processes to describe a team's behaviour.

When running the *Soccer Server*, certain options can be used to store all the data for a given match. This information is recorded in a log file, so this file is a special stream of consecutive information data. As we are considering an offline mode, the coach receives data from the log files instead of getting it from the server. Hence, from these log files, we extract important features over all the information of the field.

At every cycle of the server, each agent updates its world model with data such as positions and velocities, as well as cumulative data such total travel distances and average positions. In this first phase, the most relevant data of these log files must be extracted, so we need to get the following information:

- *Cycle:* A number that enables arrange the events.
- *Ball Position:* The ball's position is stored like *xy* axis in a coordinate system.
- *Teammate Positions:* Each teammate´s position is stored as *xy* axis coordinates.
- *Opponent Position:* Each opponent´s position.
- *Ball Possessor:* A data that indicates who is the owner of the ball.

3. 2 Event Recognition

After extracting data from the logfiles, we have to infer what events have occurred. There is some uncertainty inherent in the results because there are events very hardly to identify, even if it is done by a soccer expert. Kuhlmann et al. work [14] propose nine different events (dribble, hold, goal, pass, foul, steal, missed shot, intercepted pass and clear) to create advices in RoboCup Simulation Soccer. In our work, we only identify seven events and the way we use them is different.

One of the most important data to extract to identify high-level events is the owner of the ball every cycle. When a ball possession change occurs, it means that an event is taking place. For our research, we classify the next events:

- **PassXtoY (T):** If a player (*X*) of the team (*T*) kicks the ball and a teammate (*Y*) gains possession, then the ball owner made a pass. (Perhaps the ball owner did not want to do this pass, but we can not consider this assumption). This event stores both the player who makes the pass and the player who gains possession.

- **DribbleX (T):** If the ball moves a significant distance since the player (*X*) gains possession until he kicks it.
- **InterceptedPassXtoY (T):** If the player (*X*) kicks the ball and the opponent (*Y*) gains possession within a reasonable distance of the ball owner, the event is assumed to be an intercepted pass.
- **StealXfromY (T):** If a player (*X*) kicks the ball and an opponent (*Y*) gains possession, then the opponent stole the ball from the ball owner.
- **GoalX (T):** If a player (*X*) kicks the ball and at the end of the interval, the ball is in the goal, the event is classified as a goal by the player *X*.
- **MissedShotX (T):** If a player (*X*) kicks the ball and at the end of the interval, the ball is out of bounds, the kick is considered as a missed shot.
- **Clear (T):** If the event cannot be classified as any of the above categories.

The result of this phase is a set of events ordered by time. Each stream of events is labelled by a team (T = Left or Right) and the result may look as follows:

{Pass1to2 (R) → Dribble2 (R) → Pass2to10 (R) → Goal10 (R)}, {Pass7to10(L) → MissedShot10 (L)}

3.3 Building a Trie

Once we have extracted the features from the log files and every event have been recognized, the next step is to analyze the behaviour sequences and choose the most appropriate sequences for our goal. In this and next sections, we try to choose, as well as possible, the behaviour sequences that describe the pattern followed by the team.

As we described in section 2.2, a pattern describes a simple behaviour that a team performs. As this pattern must be predictable for the coaches, we consider that the repeating events or behaviour sequences could be related to the activated pattern. Because of this supposition, in this paper we propose the use of a trie structure data [17], [18] to store the results of the event recognition.

A trie (which is abbreviated from "retrieval") is a kind of search tree similar to the data structure commonly used for page tables in virtual memory systems. This special search tree is used for storing strings in which there is one node for every common prefix and the strings are stored in extra leaf nodes. In this research we propose to use this data structure to store events in an effective way for our goal. The advantage of this kind of data structure is that every event is stored in the trie just once, but in a way that the event has a number that indicates how many times it has appeared.

In this research, every node represents an event so a path from the root to a node represents a sequence of events in the order they were played. Works by Kaminka [15] and Huang et al. [16] use the same data structure to analyze the behaviour sequences. The goals of these two researches are: to learn the coordinated sequential behaviour of teams [15] and to create frequent patterns in dynamic scenes [16].

An example of this trie data structure is shown as follow. If we get the next events sequences from the previous phase:

{Pass1to2 (R) → Dribble2 (R) → Pass2to10 (R) → Pass10to11 (R)} and *{Dribble2 (R) → Pass2to10(R) → Goal10 (R)}* and the trie is empty, the first event sequence to insert is *{Pass1to2 (R) → Dribble2 (R) → Pass2to10 (R) → Pass10to11 (R)}* and it is added the first branch of the tree. Each event is labelled with the number 1 that indicates how many times the event has been inserted in that sequence (In Figure 1, this number is

enclosed in brackets). Then, it is inserted the two remaining suffixes of this sequence: *{Dribble2 [1] → Pass2to10 [1] → Pass10to11 [1]}* and *{Pass2to10 [2] → Pass10to11 [2]}*. Next, we insert the second sequence *{Dribble2 (R) → Pass2to10(R) → Goal10 (R)}* and the remaining suffix: *{Pass2to10(R) → Goal10 (R)}* but in this case, there exist sequences like these inserted in the trie and we just add up one to the counter number of the events.

The trie built previously for the Right team is shown in the Figure 1.

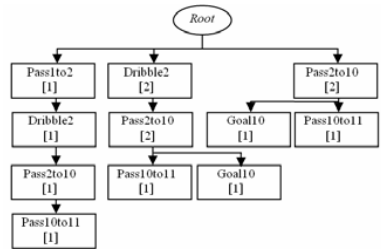

Fig. 1. Example of a trie representation

As we will see in section 5, the obtained trie from a log file game is quite big. So, in order to get the most important information and because of the pattern event sequences must be repeated to be able to extract the pattern, the tries are pruned. For pruning a trie, we eliminate the branches which have been introduced in the trie only once (it means that the node of level 1 of the branch has been introduced once).

3.4 Evaluating Dependence

Although there are few methods for discovering significance of sequences and subsequences, in this paper, we have used a statistical dependency test [16].

The main idea of this procedure is the proposal that the appearance of repetitive sequences may indicate a pattern. To evaluate the relation between the previous events sequence to a specific event (what we call prefix) and that event, we use one of the most popular statistics: Chi-square test [19]. Chi-square test is a statistical test that we propose to determine whether a prefix is dependent on the following events. This test enables us to compare observed and expected sequences objectively and evaluate whether a deviation appears. Hence, if we modify the trie structure that we have obtained in the previous section 3.3 for storing this value in every node (except the nodes of level 1 and the root), we can determine whether an event is or not relevant with its prefix.

To compute this test, a 2x2 contingency table (also known as a cross-tabulation table) has to be made. This table is filled with four frequency counts, as we can see in the Table 1. The counts are calculated as follows: The first number O_{11} indicates how many times the current node/event is following the prefix. The number O_{12} indicates how many times the prefix is followed by a different prefix. The number O_{21} indicates how many times a different prefix of the same length, is followed by the same event. The number O_{22} indicates how many times a different prefix of the same size, is followed by a different event.

Table 1. The contingency table

	Event	**Different event**	**Total**
Prefix	O_{11}	O_{12}	$O_{11} + O_{12}$
Different prefix	O_{21}	O_{22}	$O_{21} + O_{22}$
Total	$O_{11} + O_{21}$	$O_{12} + O_{22}$	$O_{11}+O_{12}+O_{21}+O_{22}$

The expected values are calculated as in *Equation 1*.

$$Expected\ (E_{ij}) = (Row_i\ Total \times Column_j\ Total) / Grand\ Total \quad (1)$$

The formula to calculate chi-squared value, is giving in *Equation 2*.

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{k} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad (2)$$

where: O_{ij} is the observed frequency and E_{ij} is the expected frequency.

4 Comparing Process

The most interesting part in our research is to provide a method to compare two agent systems' behaviours. In this research, we only compare two agent systems' behaviours, the process to compare more than two behaviours is not considered in this work.

The input of this procedure is the result of the previous process. In this procedure, we compare two different tries which represent the behaviour of two agents systems. But, one of the agent system follows an unknown pattern. The result of the comparison is a pattern description as similar as possible to the pattern followed by the pattern-trie. The term *pattern-trie* is used to refer to the trie obtained from the agents system which followed a pattern, and *no-pattern-trie* terms the trie for the agents system that did not follow the pattern.

Before describing the comparing method, we will present our proposal to store the result of the comparison: our pattern description.

4.1 Our Pattern Description

In this work, a pattern defines recurring events to a recurring prefixes. Also, a pattern could consist of a set of simple behaviours. Because of this reason, our pattern description consists of a set of sub-patterns. Our sub-pattern is defined as the possibility, measured by chi-square test value (*chi-sq*), that an event (*ev*) occurs after a prefix (*pr*). Let OurPDescription = {[p1], [p2], ...} be the set of all sub-patterns *sub-p*$_i$. A sub-pattern is defined as follow:

$$sub\text{-}p_i = (ev, pr, chi\text{-}sq)$$

Because of this is a preliminary work on the *RoboCup Online Coach* domain, we do not treat the pattern description after it is obtained. However, it will be used in future works in the online detection of possible patterns in an opponent team.

4.2 Comparing Algorithm

In this section, we describe the proposed algorithm to compare two tries (*pattern-trie* and *no-pattern-trie*) in order to get the pattern description that the *pattern-trie* follows. Before describing the algorithm, we have to consider the following concepts about the trie data structure:

- As we describe in section 3.4, every node is represented by:
  - Event: A word that indicates a specific event and the agent/s that takes part in it (*passXtoY, InterceptedPassXtoY,…*).
  - Prefix: A set of the previous events in the trie.
  - Chi-Sq: A number that indicates the chi-square test value for the node.
- The depth of a trie is the maximum depth of any of its leaves.
- The level 0 in trie is the *root*.

Also, the main features of the algorithm are:

1. A threshold value (*ThresholdChiSqValue*) has to be established for accept or reject the chi-square test hypothesis in the events.
2. If the event of a node is represented in both tries at the same level and their prefix is the same:
   - Chi-square value of both nodes is compared and only if the difference is bigger than the threshold, the information of the node of the *pattern-trie* is stored as a sub-pattern in the pattern description. It means that a different behaviour in the trie has been found (and it is classified as a sub-pattern).
3. If the event and prefix of a node are represented only in the *pattern_trie:*
   - If the chi-square value of the event is bigger than a threshold value (*Threshold-ChiSqValue*), the information of the node is stored as a sub-pattern in the pattern description.

In order to make efficient and more comprehensible the proposed algorithm, we describe, first, a few used functions:

- *depthTrie (Trie T):* This function returns the maximum depth of any of its leaves.
- *getSetOfNodes* (*Level L, Trie T):* This function retrieves a result set that contains every node of the trie *T* in the level *L*.
- *getNode* (*Event E, Prefix P, SetOfNodes S):* This function returns a node consisting of the event *E* and which prefix is *P*, and is obtained from the set of nodes *S*. (If a node with these parameters does not exist in the trie *T*, the function returns *null*).
- *chi-Sq* (*node N*): This function returns the chi-square value of the node N.
- *Prefix* (*node N*): This function returns the prefix (set of events) of the node N
- *AddToOurPatternDesc (Event E, Prefix P, Chi-Sq Chi-SqValue*): This function adds to our pattern description the new sub-pattern consisting of the event *E*, the prefix *P* and the chi-square value *Chi-SqValue.*

The basic structure of the algorithm is shown in Figure 2.

```
CompareTries (pattern_trie, no_pattern_trie)
begin
   patternDesc ← null
   maxDepth ← depthTrie (pattern_trie)

   for level_i ← 2 to maxDepth do
      {Root is considered in level 0 and the events in level 1 have no prefix}

      set_P ← getSetOfNodes (level_i, pattern_trie)
      set_NP ← getSetOfNodes (level_i, no_pattern_trie)

      for all node_p in Set_P  do
         node_np ← getNode (event(node_p); prefix(node_p); set_NP)
         if (node_np = null)
         {The node is only in the pattern_trie}

            if (chi-Sq(node_p) > ThresholdChiSqValue)
               patternDesc←AddToOurPatternDesc(event(node_p),prefix(node_p),chi_sq(node_p))
            Fi

         else  (if node_np ≠ null)

            if ( (chi_sq(node_p) - chi_sq(node_np)) > ThresholdChiSqValue)
            {The node is in both tries, so the difference between them must be considered}
               DifChiSqValue = chi_sq(node_p)- chi_sq(node_np)
               patternDesc←AddToOurPatternDesc(event(node_p),prefix(node_p), DifChiSqValue)
            fi

         fi
      od
   od
   return (patternDesc)
end
```

Fig. 2. Basic structure of the algorithm

5 Experimental Results

We carried out a set of experiments in order to test whether our method is able to compare a game played by a RoboCup soccer team which follows a pattern to a game in which this pattern is not followed. Also, the result must be a description of the detected pattern.

Robocup 2005 Coach Competition has created a set of simple strategies to be used as the base strategies of the patterns [20]. The patterns are added to theses bases strategies. A no-pattern logfile describes the base strategy and the pattern file describes a simple behaviour that the opponent team performs.

The first step in our experiment is to obtain the pattern and its corresponding no-pattern logfiles. These files, that contain the same data that the online coach receives, have been obtained from the *"Patterns"* section of *"RoboCup 2005 Coach Competition*" web page [21]. The two logfiles last around 3500 times steps (a complete game lasts 6000 times steps). The pattern and no-pattern descriptions (defined by CLang rules) are described as follows:

- Pattern Description: The attackers (i.e. players 10 and 11) pass to each other right in front of the goal, before shooting.
- No-pattern Description: The attackers (i.e. players 10 and 11) dribble to the goal and shoot (i.e. they do not pass each other).

In order to extract the main features from the log files and analyze the behaviour of the two games, we have fully implemented a program. This program is able to recognize and report the events of each game. The result is a file for each game (pattern game and no-pattern game) where is defined every event of the game. This event representation is consisting of the event realized, the player/s who made this event and the team of the player.

Then, we build a trie for each logfile with the event sequences of the opponent team (this is the only team analyzed for our purpose). The *pattern trie* represents the game in which the opponent follows a pattern and *no-pattern trie* represents the game in which the pattern is not followed. As we have seen, a trie node consists of (1) the description of an event (2) a number that indicates how many times the event has been inserted in that sequence (3) Chi-square value for this node (except for the nodes of level 1 and the root).

In this experiment, the total node number of the tries is: 176 (pattern trie) and 121 (non-pattern trie). But, as we explained in section 3.3, in order to reduce the amount of no significant nodes, the trie is pruned and the branches which have been introduced only once are eliminated.

The main characteristics of the pruned tries are:

- Pruned *pattern trie*: Total node number: 49 nodes. Depth: 6.
  - Chi-square Value: Maximum (35,0); minimum (1,3) and average (11,85).
- Pruned *no-pattern trie*: Total node number: 56 nodes. Depth: 5.
  - Chi-square Value: Maximum (25,0); minimum (0,2) and average (11,91).

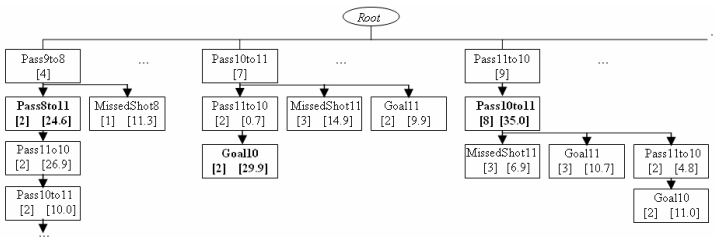

Fig. 3. Most important branches of the pattern trie

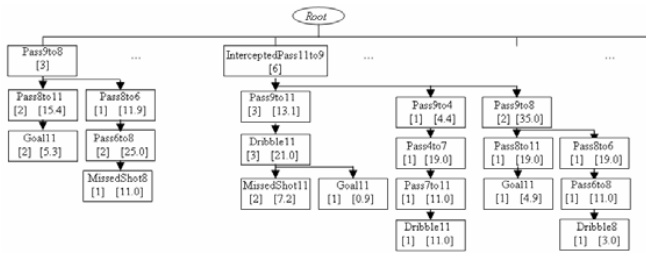

Fig. 4. Most important branches of the no-pattern trie

The most important branches of each trie are shown in Figures 3 and 4.

After obtaining the two tries that represent the two games, we can apply the algorithm described in section 4.2:

At level 2 of the pattern trie, we observe that the event “pass10to11” (and prefix “pass11to10”) does not appear in the no-pattern trie. Also, this event has the biggest chi-square value (35.0), so we insert this event and its prefix in the pattern

description. Another interesting node in this level, is the event “Pass8to11” (and prefix “Pass9to8”), but, in this case, this event appears in the no-pattern trie with a similar chi-square value, so we do not consider this event as a possible sub-pattern.

At level 3 of the pattern trie, the node “Goal10” is considered as a sub-pattern because of its chi-square value and because this node is not repeated in the no-pattern trie. So, we insert this sub-pattern in the pattern description. After analyzing every node in the pattern trie, Our Pattern Description is as follows:

OurPDescription = {[(pass10to11), (pass11to10), (35.0)],[(Goal10), (pass10to11 → pass11to10), (29.9)]}

As we can see, our pattern description represents a team behaviour very similar to the team behaviour represented by the original pattern description. So, in this case, our method is able to look for the qualitative difference between the pattern trie and its corresponding no-pattern trie. The only difference between our pattern description and the original pattern, is that the event “pass11to10” has not been identified as a sub-pattern. It is because of this event is in level 1 and at this level the nodes have no chi-square value. This improvement could be made in future works.

6 Conclusions and Future Work

In this paper we presented a comparing method of two different multi-agent systems behaviours, which one of them follows a specific pattern.

This method is applicable and useful in Coach RoboCup Soccer domain. In order to compare two different games, in which one of them is following a pattern, we use the trie data structure and it is demonstrated that a trie can be very usefulness to show the behaviour of a team in this domain.

The comparing method that we have applied works successfully when the pattern followed by a team is related to the players’ actions. But, in our research, the different field regions in which the action occurs, has not been represented, so if the pattern followed by the team is related to this aspect, our proposed method would not be viable. Also, if the pattern is related to actions that occur when the player is not the ball owner, this method, as well, would not be viable.

This is a preliminary work on this domain and we consider that the result of the proposed method is very adequate in great amount of cases.

References

1. T. L. Turocy & B. Von Stengel. Game Theory. CDAM Research Report LSE-CDAM., 2001
2. 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.
3. 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.

4. 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.
5. 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.
6. 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.
7. 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.
8. P. Riley & M. Veloso, Distributed Autonomous Robotic Systems, volume 4, chapter On Behavior Classification in Adversarial Environments, pages 371-380. Springer-Verlag, 2000.
9. T. Steffens. Feature-based declarative opponent-modelling in multi-agent systems. Master´s thesis, Institute of Cognitive Science Osnabrück, 2002.
10. 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.
11. 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, Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Sytemsi (AAMAS), Bologna, Italy, 2002.
12. P. Scerri, D. Pynadath, & M. Tambe. Adjustable autonomy in real-world multi-agent environments. In Agents-2001, 2001.
13. 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.
14. 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.
15. G. A. Kaminka, M. Fidanboylu, A. Chang & M. Veloso. Learning the Sequential Coordinated Behavior of Teams from Observations. In Proceedings of the RoboCup-2002 Symposium, Fukuoka, Japan, June 2002.
16. Z. Huang, Y.Yang and X.Chen. An approach to plan recognition and retrieval for multi-agent systems. *Proc of AORC*, Sydney, Australia. Jan 2003.
17. D. E. Knuth. The Art of Computer Programming Vol. 3, Sorting and Searching. Addison-Wesley. 1972.
18. E. Fredkin. *Trie Memory***.** Communication of the ACM**.** Vol. 3:9 (Sep 1960). pp. 490-499.
19. F.J. Rohlf, & R. R. Sokal. 1995. Statistical Tables, 3rd ed. W. H. Freeman and Company, New York, NY.
20. 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
21. http://staff.science.uva.nl/%7Ejellekok/robocup/rc05/