Classifying Efficiently the Behavior of a Soccer Team

José Antonio IGLESIAS [a] Agapito LEDEZMA [a] Araceli SANCHIS [a] and Gal KAMINKA [b]

[a] *Carlos III University of Madrid. {jiglesia, ledezma, masm}@inf.uc3m.es*
[b] *Bar-Ilan University, Israel. galk@cs.biu.ac.il*

Abstract. In order to make a good decision, humans usually try to predict the behavior of others. By this prediction, many different tasks can be performed, such as to coordinate with them, to assist them or to predict their future behavior. In competitive domains, to recognize the behavior of the opponent can be very advantageous.

In this paper, an approach for creating automatically the model of the behavior of a soccer team is presented. This approach is an effective and notable improvement of a previous work. As the actions performed by a soccer team are sequential, this sequentiality should be considered in the modeling process. Therefore, the observations of a soccer team in a dynamic, complex and continuous multi-variate world state are transformed into a sequence of atomic behaviors. Then, this sequence is analyzed in order to find out a model that defines the team behavior. Finally, the classification of an observed team is done by using a statistical test.

Introduction

Knowledge about others is very beneficial for coordination, collaboration and adversarial planning. In order to make a good decision, humans usually try to predict the behavior of others. There are new theories which claim that a high percentage of the human brain capacity is used for predicting the future, including the behavior of other humans [1]. In adversarial domains, Riley [2] supposes that human players observe the behavior of the opponent during a game and try to match the observed playing style to previously encountered ones; then, the player selects the best performed strategy against the previous opponent. Therefore, in order to act efficiently, agents (software agents, robots or human beings) should try to recognize the behavior of other agents. Different techniques have been used in agent modeling in very different areas; for instance, opponent-modeling in soccer domain simulation, intelligent user interface, and virtual environment for training. In particular, this paper focuses on the soccer domain simulation.

In this paper we propose an efficient approach for recognizing and classifying an observed team behavior. The goal of this research can be divided in 2 parts: Firstly, the different *team behavior models* (classes) are created by observing the behavior of a team during a game. Then, a team behavior is observed and classified into the predefined models (classes).

In a soccer team (multi-agent environment), a group of agents cooperates to achieve a common goal. Therefore, the changes made to the environment are not the result of the

behavior of a single agent, but the interaction of the agent with each other and the world in which they act. As a consequence, the emergent behavior is usually hard to understand because the global behavior is not the sum of the local behaviors of the agents. We propose an approach for modeling the behavior of a team as a global behavior defined by the sum of the more relevant actions of its player members.

Because of any behavior has a sequential aspect, we consider that a team actions are usually influenced by its past actions. The presented approach is an notable and successful improvement of our previous work [4]. However, in this work, a soccer *agent* team behavior is represented as a distribution of its relevant atomic behaviors. Also, an effective statistical method is used in the classification method. This approach has been fully implemented and empirically evaluated in a complex and noisy multi-agent environment: the *Soccer Server System* [5] as used in the *RoboCup Soccer Coach Simulation* [6].

The rest of this paper is organized as follows. First, Section 1 provides a brief overview of the background and related work relevant to this research. The approach is detailed in section 2. The experimental results obtained in the proposed environment are shown in Section 3. Finally, Section 4 contains future work and concluding remarks.

1. Background and Related Work

In many areas is very useful to model and recognize the behavior of other agents. Agent modeling has been applied in real-time domains with continuous state and action spaces. In competitive domains, the knowledge about the opponent can be very advantageous. For example, in the *RoboCup Simulation League* [7] the modeling and classification of the opponent team is necessary to adapt the own team behavior. In this area, Carmel and Markovitch [8] propose a method to infer a model of the opponent's strategy which is represented as a Deterministic Finite Automaton. Tambe et al. [9] present an approach for tracking agent models based on a plan recognition task. Ledezma et al. [3] present an approach to modeling low-level behavior of individual opponent agents. Time series and decision tree learning are used by Visser and Weland [10] to induce rules that describe a team behavior. Riley et al. [2] propose a classification of the current adversary into predefined adversary classes. Han and Veloso [11] recognize behaviors of robots using *Hidden Markov Models*.

Similar to our approach, Kaminka et al. [12] recognize basic actions based on descriptive predicates, and detect the relevant actions of a team using a statistical approach and a *trie* data structure. This structure is also used in [13] to create frequent patterns in dynamic scenes. However, these previous works focused on unsupervised learning, with no ability to classify behaviors into classes.

2. Soccer Team Classification. Our Approach

In this section, our approach for modeling and classifying the behavior of a soccer agent team is described. This approach is divided in two different parts: **Off-Line Phase**: For each game observed, the behavior of a team is analyzed, modeled and stored in a behavior-library (*LibTBM*) (Section 2.1). **On-Line Phase:** The goal is to classify *on-line* the behavior of an observed team into the team behavior models previously stored in the *LibTBM* (Section 2.2).

2.1. ***Off-line phase:*** *Construction of the Team Behavior Models*

The actions performed by an agent soccer team are inherently sequential, and this sequentiality is considered in the modeling process proposed. This section describes the two stages for constructing the model of a soccer agent team from its observation: First, the observation stream is transformed into an ordered sequence of recognized atomic behaviors (Section 2.1.1), then the model is created and stored in *LibTBM* (Section 2.1.2).

2.1.1. Obtaining Atomic Behavior Sequences

In this phase, each observation is a snapshot of the soccer field that do not offer any information about its actions. Therefore, the actions taken by the agent team should be estimated by contrasting consecutive snapshots. The procedure to identify atomic behaviors (events) in a soccer game is based on a work of Kuhlmann et al. [14] and eight atomic behaviors are inferred: *Pass, Dribble, Intercept pass, Steal, Goal, Missed shot, Foul* and *Hold*. Each event is characterized by the players that have executed the action. The result of this phase is a sequence of atomic behaviors ordered by time.

As a sample sequence of events, lets consider we are observing an agent team and its behavior (during a small period of time) is represented by the following sequence: $\{Pass1To2 \rightarrow Pass2To1 \rightarrow Pass1To2 \rightarrow Pass2To1 \rightarrow Dribble1\}$.

2.1.2. Creating the behavior model

In this approach, the temporal dependencies are very significant and it is considered that a current event depends on the events that have happened before and it is related to the events that will happen after. Taking this aspect into account, to get the most representative set of sequential events (subsequences) from the acquired sequence, the use of a *trie* data structure [15] is proposed. For constructing a *trie* from a single sequence of events, the sequence is processed in three steps explained below:

a. Segmentation of the sequence:

In this step, the sequence is segmented into several subsequences. This segmentation is divided by defining an appropriate length and obtaining every possible ordered subsequence of that specific length. In the proposed sample sequence, let 3 be the subsequence length, then it is obtained: $\{Pass1To2 \rightarrow Pass2To1 \rightarrow Pass1To2\}$ and $\{Pass2To1 \rightarrow Pass1To2 \rightarrow Pass2To1\}$ and $\{Pass1To2 \rightarrow Pass2To1 \rightarrow Dribble1\}$

b. Storage of the subsequences:

The subsequences of events are stored in a *trie*, such that all possible subsequences are accessible and explicitly represented. In a *trie*, every node represents an event, and the node's children represent the events that have appeared following this event. Also, each node keeps track of the number of times an event has been inserted on to it. As the dependencies of the events are relevant in an agent behavior, the subsequence suffixes (subsequences that extend to the end of the given sequence) are also inserted.

Considering the previous example, the first subsequence ($\{Pass1To2 \rightarrow Pass2To1 \rightarrow Pass1To2\}$) is added as the first branch of the empty *trie* (Figure 1 a). Each event is labeled with the number 1 that indicates that it has been inserted in the node once (in Figure 1, this number is enclosed in square brackets). Then, the suffixes of the subsequence ($\{Pass2To1 \rightarrow Pass1To2\}$) and $\{Pass1To2\}$) are also inserted (Figure 1 b). Finally, after inserting the three subsequences and its remaining suffixes, the completed *trie* is obtained (Figure 1 c).

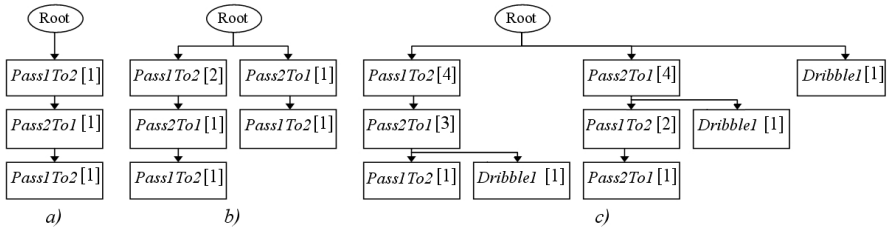

Figure 1. Steps of creating a *trie*

c. Creation of the behavior model: distribution of atomic behaviors

Once the *trie* is created, it is traversed to calculate the relevance of each subsequence (path from the *root* node to any other node of the *trie*). For this purpose, frequency-based methods are used. In particular, in this approach, to evaluate the relevance of a subsequence, its relative frequency or support [16] is calculated. Therefore, in this step the *trie* is transformed into a set of subsequences labeled with a value (support).

After labeling all the subsequences of the *trie*, the model of an agent behavior is represented by the distribution of its subsequences. In the previous example, the *trie* consists of 9 nodes; therefore, the model consists of 9 different subsequences which are labeled with its support. Figure 2 represents the distribution of these subsequences.

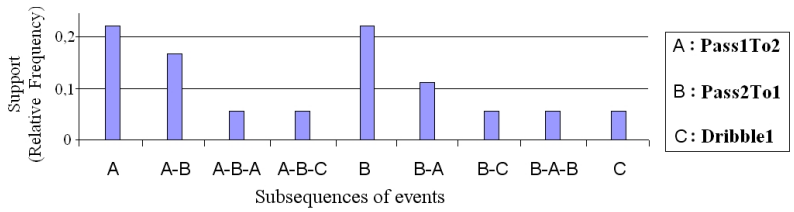

Figure 2. Distribution of subsequences.

However, in the environment in which we will apply our approach, the behavior of a team must be inferred by using two different games: *Base Strategy:* Game in which a general strategy of the team is played. And *Play Pattern:* Game in which a specific behavior of several team players is activated using the previous *Base Strategy*.

Therefore, in this case, the behavior of the team is modeled by creating the two distributions (*trie*s) of atomic behaviors of the two games (*base strategy* and *play pattern*) and then, comparing them. This comparison is needed because the relevant subsequences appears only in the *play pattern* or with a low value in the base strategy. Therefore, the result of this comparison is a distribution with the subsequences of the *Play Pattern* distribution labeled by subtracting (*Play Pattern* subsequence support) from (*Base Strategy* subsequence support).

Finally, once a behavior model has been created, it is stored in *LibTBM* as a *trie* for a good and effective handling. The different models created are stored and labeled in the library with a *name* that identifies them.

*2.2. **On-line phase:** Team Behavior Classification*

In this second phase, a game is observed and the set of behavior models that the team follows must be reported. Firstly, the observations of the agent team to classify are col-

lected and its behavior model is created. Then, this model is matched with all the behavior models stored in *LibTBM*. As both models are represented by a distribution of events, a statistical test is applied for comparing these distributions. The proposed test applied for matching two behaviors is a modification of the *Chi-Square Test* for two samples. A non-parametric test (or distribution-free) is used because this kind of test does not assume a particular population distribution. To apply this test, the behavior model to classify is considered as an observed sample and all the behavior models stored in *LibTBM* are considered as the expected samples.

The *Chi-Square Test* is the comparison of two sets of support values in which *Chi-Square* is the sum of the terms $\frac{(Obs-Exp)^2}{Exp}$ calculated from the observed (*Obs*) and expected (*Exp*) distributions (models). This test has not been used in our classification process because only the expected values are compared and if an observed value is not represented in the expected distribution, it is not considered. Therefore, in order to solve these problems, the way to compare the two distributions is modified to the sum of the terms $\frac{(Exp-Obs)^2}{Obs}$, what we call *Chi-Square-Obs Test*. By applying this test, a value that indicates the deviation of the two distributions is obtained. The lower the value, the closer the similarity between the two behaviors. Figure 3 represents graphically the idea of the proposed novel comparing method.

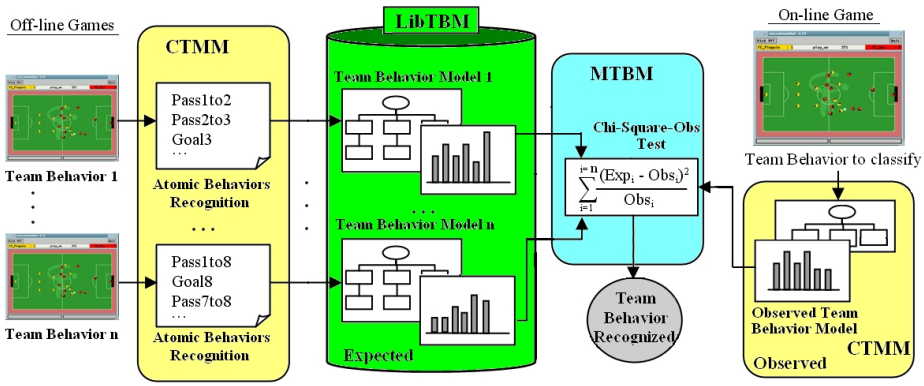

Figure 3. Team Behavior Classification Process

An important advantage of the proposed test is its rapidity because only the observed subsequences are evaluated. However, there is no penalty for the expected relevant subsequences which do not appear in the observed distribution. The *Chi-Square-Obs Test* is applied once for each behavior model stored in *LibTBM*. The number of terms to sum in each comparison is always the same: number of subsequences in the observed behavior model. It means that the degrees of freedom (*dof*) are the same in all the comparisons and a normalization of the results according to the *dof* is not necessary. The model which comparison obtains the lowest value is considered as the most similar one.

As example, lets consider the sequence that represents the observed behavior is: $\{Pass1To2 \rightarrow Pass2To1 \rightarrow Goal1\}$. Figure 4 shows the comparison between the previous expected distribution (*Expected Team Behavior Model 1*) and the observed distribution (*Observed Team Behavior Model*). The comparison value in this example is: $\frac{(0,22-0,16)^2}{0,16} + \frac{(0,16-0,16)^2}{0,16} + \frac{(0,22-0,16)^2}{0,16} + \frac{(0-0,16)^2}{0,16} + \frac{(0-0,16)^2}{0,16} + \frac{(0-0,16)^2}{0,16} = 0{,}525$.

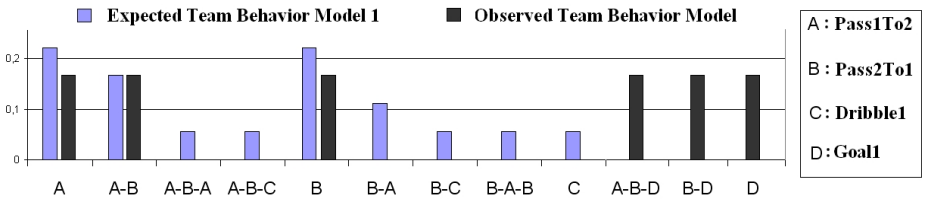

Figure 4. Observed and Expected Comparison Example

3. Experiments. RoboCup Soccer Team Behavior Classification

The *Soccer Server System* [5] is a server-client system which simulates a soccer game in a discrete fashion. Eleven players (agents) can only perceive objects that are in their field of vision and interact in a complex and noisy multi-agent environment. In particular, this research uses the environment of the *RoboCup Soccer Coach Competition* [6].

3.1. RoboCup Soccer Coach Simulation

The main goal in this environment is to classify a soccer simulation opponent team behavior by observing its actions (the behavior of the players does not vary significantly over the course of the game). The construction of models is done by analyzing several game *log files*. Each game represent a *team behavior*. Then, it is observed a new game in which several *team behaviors* are activated at the same time. The classification is done by reporting the *team behaviors* activated in the observed game.

For the experiments in this domain, we have used the rules from the *RoboCup 2006 Coach Competition* [17] and the experiments have been performed in the same way that this competition. However, although the competition consists of 3 different rounds with different *team behaviors* to analyze and classify, we only show the results of the first two rounds obtained using our approach. In the first round, 17 different *team behaviors* are analyzed (download from [6]) and stored in *LibTBM*. The games analyzed in this case are around 1000 to 3000 cycles games (in our case, the atomic behaviors identified for each game are around from 100 to 200). Then, in each iteration of the round, a different game where 4 or 5 different *team behaviors* have been activated is observed (*team behavior*).

3.2. Results

Table 1 shows the ranking list obtained for the 3 iterations of the first and second rounds (because of lack of space, only the 8 first *team behaviors* have been represented). The number of *team behaviors* activated in each iteration is indicated in brackets. In the first round, the *team behaviors* are identified with a number (from pattern00 to pattern16) and in table 1 the *team behaviors* that have been activated are marked with an asterisk. As we can see, the result are very promising since in the six first places of the 6 iterations, there are at least 3 *team behaviors* that have been correctly identified as activated.

Analyzing these results, the *player behaviors* named *pattern04* and *pattern16* have been classified correctly in first position. Although the way to define these *player behaviors* is using a special language called *CLang* [18] (with which the behavior of the simulated soccer player can be modified), we describe these *player behaviors* as follows:

Table 1. Results for the *RoboCup Coach Competition*. Round1

Round1 (17 different team behaviors)			Round2 (16 different team behaviors)		
Iter1 (4)	*Iter2* (5)	*Iter3* (5)	*Iter1* (5)	*Iter2* (4)	*Iter3* (4)
pattern04 *	***pattern16 ****	***pattern04 ****	***patternJ ****	patternB	patternF
pattern16	***pattern01 ****	***pattern02 ****	patternL	patternF	***patternE ****
pattern00 *	pattern00	pattern13	***patternF ****	patternA	patternJ
pattern12	***pattern13 ****	pattern05	patternE	***patternE ****	patternD
pattern15 *	pattern05	***pattern00 ****	patternD	patternJ	***patternB ****
pattern05	***pattern07****	***pattern12 ****	***patternA ****	***patternP ****	patternH
pattern09	pattern03	pattern01	patternG	patternD	patternA
pattern03	pattern09	***pattern06 ****	patternP	***patternK ****	patternP
...	...	...	...	...	...

- ***Pattern04:*** *Players 6, 7 and 8 pass the ball to an specific point in the field.*
- ***Pattern16:*** *Player 1 pass to player 3 or 4. Player 3 and 4 dribble to a fix space.*

However, the following *player behaviors* are not recognized correctly:

- ***Pattern14:*** *The player 3 dribbles to the space where the player 5 is situated. The player 5 dribbles to the space where the player 9 is situated. The player 9 dribbles to the space where the player 3 is situated.*
- ***Pattern08:*** *If the ball is situated in a defined area, player 8 dribbles to a fix space. Otherwise, player 8 passes to player 0.*

As we can see analyzing the results, our approach works successfully when the behavior of the team to recognize is related to the actions of the players. Any other kind of *team behavior* (related to the different field regions in which the action occurs or the cycle when it occurs) could not be detected. However, there are some *team behaviors* that are not related to actions but the way the players play makes that it can be recognized.

These results are very interesting for the official goal of the *RoboCup* (*to create a team of fully autonomous humanoid robot soccer players*) because, as humans do, the robot players should recognize the opponent behavior in order to act optimally.

4. Conclusions and Future Works

This paper presents an approach for modeling and classifying a soccer team behavior from observation based on a previous work [4]. The underlying assumption in this approach is that the observed team behavior can be transformed into a sequence of ordered atomic behaviors. This sequence is segmented and stored in a *trie*, then the relevant subsequences are evaluated by using a frequency-based method. The main aspect in this approach is that the model of an agent behavior is represented by a distribution of relevant subsequences. Also, for classifying a given team behavior, a modification of the *Chi-square Test* for two samples is proposed.

This approach has been evaluated in the real-time and multi-agent domain *RoboCup Soccer Coach Simulation* by conducting a large set of experiments. Although the results

depend of the behavior to recognize, the obtained results by using the proposed approach are very satisfactory. The approach is evaluated only in one domain; however the only step of the approach which is domain-dependent is the transformation from observation to a sequence of atomic behaviors. Therefore, the approach can be generalizable to modeling and classifying other behaviors represented by a sequence of events (such as GUI events, network packet traffic and so on).

In this research, we have considered that the behavior of a team does not change over a game. However, in order to construct a soccer team model more realistic, it should be frequently revised to keep it up to date. This aspect could be solved by using *Evolving Systems* and it is proposed for future work. The use of the classification result for carrying out useful actions in the proposed environment is also considered as future work.

Acknowledgments. This work has been supported by the Spanish Ministry of Education and Science under project TRA-2007-67374-C02-02.

References

[1] N. J. Mulcahy and J. Call. Apes save tools for future use. *Science*, 312(5776):1038–1040, May 2006.

[2] P. Riley and M. M. Veloso. On behavior classification in adversarial environments. In *DARS*, pages 371–380.

[3] A. Ledezma, R. Aler, A. Sanchís, and D. Borrajo. Predicting opponent actions by observation. In *RobuCup*, volume 3276 of *Lecture Notes in Computer Science*, pages 286–296. Springer, 2004.

[4] J. A. Iglesias, A. Ledezma, A. Sanchis. A comparing method of two team behaviours in the simulation coach competition. InV. Torra, Y. Narukawa, A. Valls, J. Domingo-Ferrer (Eds.), MDAI, Vol. 3885 of LNCS, Springer, 2006, pp. 117–128.

[5] I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: a tool for research on multiagent systems. In *Applied Artiificial Intelligence*, volume 12, pages 233–258. Taylor and Francis Ltd, 1998.

[6] The robocup 2006 coach competition web page (http://ssil.uni-koblenz.de/rc06/), December 2006.

[7] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge 97. In *IJCAI97*, 1997.

[8] D. Carmel and S. Markovitch. Opponent modeling in multi-agent systems. In *Adaptation and Learning in Multi-Agent Systems*, pages 40–52. Springer-Verlag: Heidelberg, Germany, 1996.

[9] M. Tambe and P. S. Rosenbloom. Architectures for agents that track other agents in multi-agent worlds. In *ATAL*, pages 156–170, 1995.

[10] U. Visser and H.-G. Weland. Using online learning to analyze the opponent's behavior. In *RoboCup*, pages 78–93, 2002.

[11] K. Han and M. Veloso. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of IJCAI-99 Workshop on Team Behaviors and Plan Recognition*, 1999.

[12] G. A. Kaminka, M. Fidanboylu, A. Chang, and M. M. Veloso. Learning the sequential coordinated behavior of teams from observations. In *RoboCup*, volume 2752 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2002.

[13] Z. Huang, Y. Yang, and X. Chen. An approach to plan recognition and retrieval for multi-agent systems. In *Proceedings of AORC*, 2003.

[14] G. Kuhlmann, P. Stone, and J. Lallinger. The champion UT Austin Villa 2003 simulator online coach team. In *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.

[15] E. Fredkin. Trie memory. *Comm. A.C.M.*, 3(9):490–499, Sept. 1960.

[16] R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995.

[17] R. Committee. Robocup 2006 official rules for the competition (http://ce.sharif.edu/ m_sedaghat/ robocup/orga/rc06/rules0.0.pdf, December 2006.

[18] M. Chen, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, and X. Yin. Soccerserver manual ver. 7, 2 2002.