Feature-Based Declarative Opponent-Modelling

Timo Steffens

Institute of Cognitive Science, Osnabrueck, Germany
timosteffens@gmx.de
http://www.cogsci.uos.de/~tsteffen

Abstract. In the growing area of multi-agent-systems (MAS) also the diversity of the types of agents within these systems grows. Agent designers can no longer hard-code all possible interaction situations into their software, because there are many types of agents to be encountered. Thus, agents have to adapt their behavior online depending on the encountered agents. This paper proposes that agent behavior can be classified by distinct and stable tactical moves, called features, on different levels of granularity. The classification is used to select appropriate counter-strategies. While the overall framework is aimed to be applicable in a wide range of domains, the feature-representation in the case-base and the counter-strategies is done in a domain-specific language. In the RoboCup domain the standard coach-language is used. The approach has been successfully evaluated in a number of experiments.

1 Introduction

An important factor effecting the behavior of agents in MAS is the knowledge which they have about each other [3]. Yet, in open domains like RoboCup agents encounter a variety of opponents which are not known to them beforehand. Traditional methods for inferring the plans or actions are not always applicable. Plan recognition [4] [5] focusses on deliberative agent architectures, so its application in dynamic domains in which (partly) reactive agents are to be encountered is not successful. Tambe attacked the problem of rapidly changing plans, but relies on several simplifying assumptions, e. g. that modelling and modelled agent share the same operator hierarchy and that the exact internal state of the modelled agent is known [10].

A promising approach which this work builds upon is to compare the tactics of new opponents to previously encountered ones and then select a strategy which has been successful back then [7]. It has been successfully applied in setplay-situations [8].

This work extends this case-based-reasoning approach by applying it to other game situations as well, and more importantly, by introducing a more flexible and more compact way of representation.

The remainder of this paper is organized as follows. Section two defines the representation formalism, feature-based opponent models. Section three describes how this framework can be applied in RoboCup and shows first experimental results. Finally, section four concludes.

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 125–136, 2004.

2 Feature-Based Models

This section defines features and feature-based models in a domain-independent way. Also an architecture for agents employing this approach is proposed.

2.1 Features

The general idea is that humans use only the most typical properties of the opponent's behavior to retrieve a similar known opponent strategy. The assumption is that a limited number of opponent models can describe a wide range of opponents [8]. While Riley stores every observation in the opponent-models [7], feature-based models contain only a small number of distinct and stable features. An example of such a feature for the RoboCup domain:

```
The opponent often does long passes along the left wing
to the forwards.
```

And for the MODSAF domain [2], a military air-combat simulator:

```
If the aircraft comes into radar range, it turns onto
collision course.
```

We propose that rules which map actions to situations are the proper means to express such features. It is obvious that the features depend on domain-specific concepts like passing or radar-range. So a domain-specific language is necessary which can formalize situation and action descriptions. In RoboCup this can be accomplished by the standard coach language (CLang) [6].

Definition 1. *Let S be the set of all situation descriptions. Let H be the set of all possible actions. Let $H*$ be the powerset of H. A feature $< s, A >, s \in S, A \in H*$ is a mapping from a situation description s to a set of actions A.*

On this abstract level, situation descriptions may range from raw sensor data to preprocessed conceptualizations with domain-specific concepts. The situation descriptions in S may be incomplete, thereby focussing on only some part of the world. The actions in H are assumed to carry information on which of the agents executes it. As an example, in RoboCup a pass from player 6 to player 11 is different from a pass from player 10 to 11.

As suggested by the above examples, features are of a probabilistic nature. Teams may only tend to execute certain moves to a certain degree of probability. So features may even contradict each other, thereby reflecting the team's ability to decide between alternative options from their repertoire. This is treated in the feature-based declarative opponent-model (FBDOM):

Definition 2. *A feature-based declarative opponent model ω is a set of 2-tupels (F_i, p_i), where the F_i are features and the p_i are probabilities of the features to occur in the strategy which is specified by the model.*
Also the constraint $(s_i = s_j \wedge A_i = A_j) \rightarrow p_i = p_j$ has to be satisfied.

Two aspects of features need to be satisfied before a human deems them typical for a strategy. These are distinctness and stableness which originate from image

recognition [12]. Distinctness means that a feature appears only seldom, but if it does the probability of a certain class is high. A stable feature on the other hand appears with a high probability in the class. Not identical notions, but close approximations to them, are used for building the opponent models. In this work, distinctness of a feature f is assumed if $p(M|f) > \alpha$, where M is an opponent model and α is a manually set threshold. Stableness is checked by $p(f|M) > \beta$, where β is another manually set threshold.

2.2 Architecture for FBDOM

This section outlines an architecture for an agent applying FBDOM. As shown in figure 1, the basis are the opponent models containing tactical descriptions on different levels of granularity and specificity. E. g., some might specify the complete strategy of a certain team, others might only specify the marking-assignment of the left-wing defender. In order to detect the features in the opponent models, the observations which come in as raw sensor data have to be processed by the action- and situation-detectors. They try to match the observations into the features of the models. The information if and how the observations match will be passed on to the model selector. It handles for example cases which match only partially (see below for a discussion of the different matching methods), and implements one of the possible selecting methods, e. g. a Bayesian classifier, or Tversky's contrast model for similarity [11]. The opponent model with the best value will be chosen and then a knowledge base will decide which counter-strategy is applicable. Just like the opponent models, the counter-strategies can vary in size. They can either contain full team strategies or just partial specifications, e. g. how a forward should shoot.

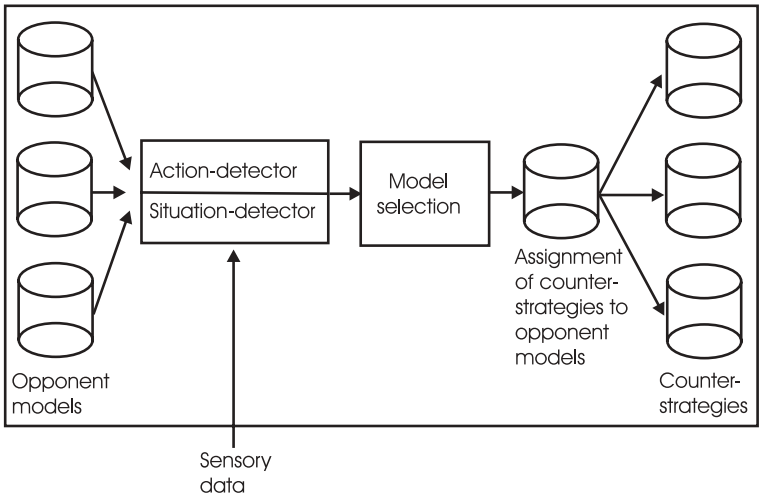

Fig. 1. Architecture of a FBDOM system.

This architecture allows usage as an online- or offline-method. It depends on the domain though, if the method needs only so much data as is provided during the encounter with other agents, or if the histories or logfiles of previous encounters have to be analyzed in order to select the counter-strategy beforehand.

3 Application in RoboCup

This section provides a proof of concept for FBDOM by implementing it in the RoboCup simulation league domain. The advantages and shortcomings of this approach are highlighted, and a series of experiments is described.

3.1 Representing Features for RoboCup

The performance of FBDOM is expected to be highly dependent of the used representation language. It has to be expressive enough to cover a variety of situations and actions, and should provide different levels of granularity (e. g. ranging from general descriptions of pass regions of a team to more specific passing behaviors of a particular player).

For the experiments, the standard coach language (CLang) [6] was chosen, because it fulfills these demands. Initially, CLang was designed to let the online coach inform and advice its field players. But it is also suited to represent strategies, since its messages are basically production rules, mapping situations to actions. The situations are crisp (possibly incomplete) descriptions of the world state like player and ball positions, play modes, score, and time, combined by logical operators. The actions are highlevel-actions which are more abstract than the server primitives and include concepts such as passing-to-regions, passing-to-players, positioning, and marking. As an example, the first example for a feature given before would be expressed in CLang as follows:

```
(definerule rule1 direc
  (and (bowner opp {X}) (bpos BACK_LEFT_WING))
  (do opp {X} (pass FRONT_LEFT_WING)))
```

The first line is due to the coach protocol and irrelevant here. The second line is the situation description and denotes that the opponent has the ball and that the ball is in a specific region (which is defined elsewhere). The action is specified in the last line and says that the ballowner does a pass to another region.

3.2 Situation-Matching

Since the CLang situation concepts are externally observable (i. e. they do not rely on hidden or internal states), the situation descriptions of the features can be matched easily to the actual worldstate. In fact, coachable teams have to implement situation-matching in order to determine when coach advice is applicable to a situation. The situation-matching code for these experiments was slightly adapted from the Dirty Dozen [1]. Although CLang conditions are made up of logical operators and freely definable regions, situation-matching is decidable. It can be seen as checking the includedness of points (concrete observed situations) into regions (general conditions) in the state space.

3.3 Action Detection

While the situation-matching is straight-forward, detection of actions is highly ambiguous. E. g. for an external observer it is hard, if not impossible, to decide if a kicked ball was passed to a player on the ball's trajectory or if it was shot to a certain point on that line. Thus, observed actions are often not singular points in the state-space, but regions. To match actions, we chose an expectation-driven approach. That is, observed actions are matched successfully to a feature's action, if any interpretation of the observed one is not mutually exclusive to the feature's action. To illustrate this:

Consider a ball that was kicked and travels along a certain line. Although the intention of the shooter is not clear, its action would match any feature's action that describes a shot or pass to any point or player on that line. That is, any overlap in situations is considered a match. For a more detailed discussion about operationalizations of CLang actions see [9].

3.4 Building the Models

Having accurate models for different opponent classes is crucial. The models of FBDOM are more complicated to build than those of Riley's approach. While Riley's models can be built by just counting positional observations of the modelled teams [8], the models here need distinct and stable features. That is, building such a model is basically about finding features that reliably describe the behavior of opponents that belong into the class.

Up to now the models are defined by a domain expert. By definition, features are typical moves. For the experiments opponent models for the offensive behavior of five teams were created manually. Surprisingly, it was sufficient to watch one or two games in order to build a model for a given team, because the typical, characteristic behaviors of a team have high saliency. The number of features in the models ranged from two to fifteen. The associated probabilities were then acquired automatically by determining the frequencies in the modelled team and in other teams. These frequencies were also used to ensure distinctness and stableness. If the frequency of a feature f in a model ω of team t was not significantly greater than the frequency of f in all other teams, f was removed from ω, because it was not distinct enough for t. f was also removed if its frequency was not beyond a certain threshold, in order to ensure stableness. In fact, only three features had to be removed (two due to indistinctness and one due to instableness), owing to the expertise of the domain expert. Still, future work aims at acquiring features and models by clustering or rule learning.

3.5 Determining the Best Matching Opponent Model

Feature-based models do not explain every observation, but just a subset. This has to be taken into account when determining the best matching model. Several methods are possible and need to be evaluated.

Matching Types: In MAS pairs of actions can be mutually exclusive (mutex). This depends on the domain, e. g. in RoboCup an agent cannot turn and kick at the same time. While in simple one-agent-domains all pairs of action may be mutex, in MAS most of the action pairs involving two agents are not mutex.

Definition 3. *Two actions a and a' are* `mutually exclusive` *or short* `mutex` *to each other, if they cannot be executed simultaneously. For two mutex actions a and a' we write $a \div a'$.*

Definition 4. *An action a is* `mutex` *to a set of actions A iff $\exists a' : a' \in A \wedge a \div a'$. We write $a \div A$.*

Definition 5. *A situation s is* `subsumed` *by situation s', iff s is true whenever s' is true. We write $s \subset s'$.*

Definition 6. *Let S be the set of all situation descriptions. Let H be the set of all actions. Let $H*$ be the powerset of H. An* `observation` *is a tupel $< s, A >$, $s \in S$, $A \in H*$, where s is the description of a complete observed situation and A is the set of observed actions in that situation.*

Since observations may contain several typical moves at once (say a defender stays in a certain region while the forward passes along a typical line), not observations, but features have to be counted. A feature $< s', A' >$ can be evaluated into the following primitive results wrt. an observation $< s, A >$. In the following list of match-types the probability parameter of the features in the model is abandoned, having no influence on matching. Because two actions are either mutex or not, this list is complete:

- `No-match`: $\neg\ (s' \subset s)$
  The situation of the feature in the model does not match the observation.
- `Partial Match`: $s' \subset s\ \wedge\ \exists a : (a \in A'\ \wedge\ a \in A)$
  The situation matches and there is at least one shared action in the feature action set and the observation action set.
- `Full Match`: $s' \subset s\ \wedge\ \forall a :\ (a \in A'\ \rightarrow a \in A)$
  The situation matches and all actions of the feature are in the observation action set.
- `Partial Mismatch`: $s' \subset s\ \wedge\ \exists a :\ (a \in A'\ \wedge\ a \div A)$
  The situation matches and there is at least one action in the feature action set which is mutex to the observation action set.
- `Full Mismatch`: $s' \subset s\ \wedge\ \forall a :\ (a \in A' \rightarrow\ a \div A)$
  The situation matches and all actions in the feature action set are mutex to the observation action set.

Partial match and partial mismatch can apply together within a given feature. In the selection process, partial matches or mismatches are ignored. On the other hand, a given observation can result in any type for different features, e. g. in a full match in feature 1 and a full mismatch in feature 2. This has to be taken into account when comparing opponent models in terms of the number of matches.

Selection Parameters: The above considerations lead to several parameters that can be combined and need to be evaluated:

- The first parameter (Once vs. All) determines how many matches will be counted for each situation. In case of "Once", the matching process aborts after the first successful full match. In the other case several features may be matched in the same situation. "Once" also means that a match overrules a mismatch of another feature.
- The second parameter (Most vs. Ratio) triggers whether the model that has the highest number of matches or the one that has the best match-mismatch ratio will be selected (cf. Tversky's ratio model [11]).
- The third parameter (Increasing vs. Normalized) specifies if the number of full matches and full mismatches will be divided by the number of features in the model. This way a normalization is done to overcome the variability in the number of features.

Combining these parameters results in eight different selection methods which were evaluated and compared to the Bayesian classifier, which is one of the most common methods in feature-based approaches [12]:

$$P(M_i|\alpha) = \frac{P(\alpha|M_i)P(M_i)}{P(\alpha)}$$

where M_i are the models and α is the observation.

3.6 Benefitting from the Classification

Of course a classification of the opponent alone does not improve the team's performance. This knowledge about the opponent's behavior must be exploited. So for each model a counter-strategy has been created manually. The counter-strategies were built depending on the characteristics of the model. To illustrate this, some examples are listed:

- If the model specified a fixed formation, a counter-formation was used. I. e. in defense players pool around the opponent's forwards and offensive midfielders, and in offense the forwards are located in the free spaces of the opponent's defenders.
- If the positions of the forwards were variable, but the forwards kept their role throughout the game, then the defenders were assigned marking assignments.
- If a model used fixed setplays (positions and/or pass chains), the counter-strategy incorporates marking assignments or positions for kick-offs based on the opponent's positions etc.

For the further experiments it was important to test the counter-strategies' appropriateness against the modelled teams. This was tested by feeding the counter-strategies into the Dirty Dozen (DD) team whose behavior is specified by an extension of CLang. So the counter-strategies are directly executed [1].

It turned out that the counter-strategies indeed achieved significantly higher scores against the modelled team than the baseline strategy which was used by DD during RoboCup 2001 (see [9] for the statistical values). So the strategies could be used in the following experiments.

3.7 Experiments

The experiments were designed to test if feature-based models are able to represent opponent behaviors, if they generalize to previously unseen teams, and what effect the observation length (i. e. the amount of classification data) has. Although FBDOM can be done decentralized, the classification was done by a centralized coach agent in these experiments.

Experiment 1. Considering that the models were built from only one or two games per team, the first experiment had to verify that the models are able to code the behavior of the team they were built for in new games and against other opponents. Additionally, the experiment was used to determine the best parameter settings (cf. section 3.5).

There were five opponent models $OM_1 - OM_5$ which had been built for teams $T_1 - T_5$. Now each team played several times against all other teams, including itself. 20 games thus resulted in 40 test instances for the nine counting methods. A classification was counted as correct, if OM_i was selected for team T_i. To fare better than random, more than 20% accuracy had to be achieved. Interestingly all normalized methods performed better than the increasing ones (see table 1). All normalized methods and the best increasing method were significantly better than random ($p < 0.01$), showing that the models were able to generalize to new games against new opponents. Especially one parameter setting achieved an accuracy of 82.5 %, which is a promising result for the following experiments.

Table 1. Parameters for counting methods and their identification success.

Normalized	**Once**	**All**
Most	82.5 %	80 %
Ratio	75 %	65 %

Increasing	**Once**	**All**
Most	25 %	35 %
Ratio	32.5 %	47.5 %

Bayes
26 %

Surprisingly, the Bayesian classifier performed on random niveau. A possible explanation for this is that the inter-dependence of the features violated the independence demand of Bayes. For example, a feature saying that a forward shoots from the left wing on the goal is highly dependent on a feature that specifies that the midfielders pass the ball on the left wing to the forward. These feature dependencies are likely to render Bayesian classification unsuccessful.

Experiment 2. In order to test if the team can benefit from the models even if it plays against a totally new opponent, experiment 2 had to test if classifying

a new opponent and then playing with the appropriate counter strategy yields better goal-differences than playing with the baseline strategy, which was the behavior specification of DD as it competed at RoboCup 2001.

There were six new teams $N_1 - N_6$, the baseline strategy DD, the opponent models $OM_1 - OM_5$, and the counter strategies $CS_1 - CS_5$. For each new team N_j a baseline goal-difference was found by running several games against DD. In order to use the logfiles effectively, these games were also used to classify the new teams. Subsequently, according to each single classification OM_i, the new team played against the counter strategy CS_i. This resulted in two sets of games per new team with exactly the same number of instances, the baseline games (set 1) and the counter strategy games (set 2). Note that in the course of experiments some games had to be removed from consideration because of server crashes or connection errors (see N_1 and N_2 in table 2). The results show that in five out of six cases the classification and the related counter-strategy yielded significantly better scores than the baseline.

Table 2. Goal-difference mean and number of games of the new teams against baseline and selected counter-strategies. p is depicted if difference is significant (1-tailed t).

Team	Mean of counter-strat.	N_1	Baseline mean	N_2	p
Cyberoos 2000	-4.694	172	-5.125	183	/
FC DrWeb 2001	-5.46	126	-6.89	128	0.05
Helli Respina 2001	-13.84	72	-18.19	83	0.05
Virtual Werder 01	-0.839	30	-1.375	31	0.05
Mainz Rolling Brains	-13.21	23	-21.08	23	0.025
FC Portugal 2001	-34.27	21	-44.58	23	0.005

Experiment 3. While the significant better goal-differences against the classified counter-strategies in experiment 2 are a necessary condition for showing that the method is successful, they are not a sufficient condition. It might still be the case that all counter-strategies are better in general than the baseline strategy. Especially since the scores are partly very negative, the improvements might be due to a floor effect. In such a case, the classification would be obsolete, because any choice between the counter-strategies would yield better results than using the baseline strategy. So another experiment was run, in which several games were run against the new teams in which the strategy was randomly selected. That is, DD played against team N_i by using randomly selected counter-strategies CS_j. If the means of these games are less than the means of experiment 2 in which the classified counter-strategy was used, it can be assumed that the optimal strategy was used in the classification & selection-runs. The outcome of this experiment was non-uniform (see table 3). In three cases there was no significant difference between the random selection and the selection based on classification. One of those teams had not yielded significantly better results in experiment 2, so this case was not surprising. At least in the three other cases, the selection based on classification performed significantly better than the random selection. This means that in these cases the most suitable opponent model

Table 3. Goal-difference mean of games in which the strategy was selected randomly and of games, in which classified strategy was used. Differences to values in previous tables are due to an increased number of games.

Team	random	N_1	classif.	N_2	α
Cyberoos 2000	-4.77	126	-4.69	172	/
FC DrWeb 2001	-5.50	168	-5.85	163	/
Helli Respina 2001	-16.68	173	-14.25	173	0.05
Virtual Werder 01	-1.352	54	-0.8197	61	0.025
Mainz Rolling Brains	-15.65	336	-14.11	343	0.1
FC Portugal 2001	-32.50	42	-33.96	45	/

and the corresponding counter-strategies were selected in experiment 2. It also means that the opponent-models and counter-strategies did indeed generalize over the new teams, making the FBDOM approach successful.

However, the three cases in which the approach was not better than random selection have to be discussed. One possible explanation is that the strategies of the involved teams are so similar, that the related counter-strategies perform similarly well. Interestingly, two of these three cases were classified as ATTCMU or FCPortugal most of the times (see [9] for details), which are very similar to each other anyway, even for human observers. From this it can be concluded, that also the counter-strategies for ATTCMU and FCPortugal might perform similarly, which would contribute to the lack of significant difference. Another reason for the outcome that several counter-strategies performed similar might be that they are only different in the defensive parts, because the opponent models focussed on offensive behaviors. So, whenever the team was in a defense situation, there was no difference between any of the counter-strategies. Anyway, in the three cases which did not achieve significant improvements, the counter-strategy that was selected by the classification was not better than the others. This might also be due to the fact that the five created opponent-model/counter-strategy pairs cannot be assumed to cover all existing teams. As of now there is no evidence how well the six new teams are covered by the opponent-models. Based on this last thought it is strong evidence for the quality of FBDOM, that three cases were nevertheless significantly better than the baselines.

Experiment 4. In order to test the amount of data needed for the classification, six observation lengths were tested. The same recorded games and settings as in experiment 1 were used, with the difference that only the best parameter setting was used and that the observation length was variied. All observations started at kick-off. The results show that the classification performs very well even for very short observation windows (see table 4). After 100 cycles the classification is significantly ($p < 0.05$) better than random selection (which is 20%). After 250 cycles the classification is already correct in more than 50% of the cases. The accuracy gets better the more data is acquired.

Table 4. Accuracy of selection for different lengths of observation window.

Intervall length	accuracy
100	42.5 %
250	57.5 %
500	62.5 %
1000	62.5 %
3000	67.5 %
6000	82.5 %

This means that the classification cannot only be done offline by analysing logfiles, but also online. This also renders the idea applicable to select rather general models in the beginning of the game, and select more detailed models when more data is acquired.

4 Conclusion

A method for representing opponent models in multi-agent-systems was introduced and its performance was experimentally evaluated in the RoboCup domain. It was claimed that for classifying an opponent it is sufficient to focus on distinct and stable features instead of processing the complete behavior for all situations. The assumption was that a set of opponent models covers a great amount of existing opponent behaviors. The experiments showed that the identification accuracy was high for the modelled teams, so the claim can be supported that features are a well-suited method to describe opponent behaviors.

Regarding the coverage of new teams, the experiments were non-uniform, but hint in a promising direction. There are some methodological difficulties to measure the impact of counter-strategies. In a perfect experimentation setting, each created counter-strategy would perform well against only one opponent model, and bad against all other models. Yet, this can only be achieved in restricted toy-domains or against manually created opponents, but not under realistic conditions with using real teams. Obviously in the experiments the five created opponent models were not enough to cover all new teams. In five of six cases, the selected counter-strategy performed better than the baseline, and three of these five cases were also better than random selection. More work is needed to verify that the cause for the unsuccessful cases was the similarity between the opponent models and the small number of models which cannot be expected to generalize over all new teams. Creating more elaborate models that also contain information about defensive situations or in-depth analysis of the existing offensive behaviors could be helpful for this further work.

However, features form compact opponent models which successfully generalized over several new teams, so that the related counter-strategies were effective against previously unknown opponents. This also revealed that tactics can be identified by certain typical features, which are at this state of RoboCup still independent of the opponent, as the experiments suggest. Because of this, opponent models can easily be created for a team by observing arbitrary opponents playing against that team.

Acknowledgements

Thanks to Claus Rollinger and Wilfried Teiken for many useful comments.

References

1. S. Buttinger, M. Diedrich, L. Hennig, A. Hoenemann, P. Huegelmeyer, A. Nie, A. Pegam, C. Rogowski, C. Rollinger, T. Steffens, W. Teiken: The Dirty Dozen Team and Coach Description. In: A. Birk, S. Coradeschi, S. Tadokoro, editors: RoboCup-2001: Robot Soccer World Cup V, Springer, Berlin, 2002
2. R. B. Calder, J. E. Smith, A. J. Courtemarche, J. M. F. Mar, A. Z. Ceranowicz: Modsaf behavior simulation and control. In Proceedings of the 2nd Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, 1993
3. David Carmel, Shaul Markovitch: Incorporating Opponent Models into Adversary Search. Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press, Portland, Oregon (1996)
4. Gal Kaminka, D. V. Pynadath, Milind Tambe: Monitoring Deployed Agent Teams. Proceedings of the Fifth International Conference on Autonomous Agents (Agents-2001), Montreal, Canada (2001)
5. A. Kautz, F. Allen: Generalized plan recognition. In Proceedings of the National Conference on Artificial Intelligence, pp. 32-37, AAAI Press, Menlo Park, CA, 1986
6. M. Chen, E. Foroughi, F. Heintz, Z. X. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Yi Wang, and Xiang Yin. Soccerserver Manual v7. RoboCup Federation, 2001.
7. Patrick Riley, Manuela Veloso: On Behavior Classification in Adversarial Environments. In Lynne E. Parker, George Bekey, Jacob Barhen (eds.): Distributed Autonomous Robotic Systems 4, Springer, Heidelberg, Germany, pp. 371-380 (2000)
8. Patrick Riley, Manuela Veloso: Planning for distributed execution through use of probabilistic opponent models. In Proceedings of the IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information, 2001
9. Timo Steffens: Feature-based Declarative Opponent-Modelling in Multi-Agent-Systems. Master thesis, University of Osnabrueck (2002)
10. Milind Tambe, Paul S. Rosenbloom: Architectures for Agents that Track Other Agents in Multi-Agent Worlds. In M. Wooldridge, Joerg P. Mueller, M. Tambe (eds.): Proceedings on the IJCAI Workshop on Intelligent Agents II : Agent Theories, Architectures, and Languages, Springer, Heidelberg, Germany, pp. 156-170 (1996)
11. A. Tversky: Features of similarity, Psych. Review, 84(4), July 1977, pp. 327-352
12. Paul Viola: Complex feature recognition: A bayesian approach for learning to recognize objects. Technical report AIM-1591, AI Lab, MIT, 11, 1996