From Basic Agent Behavior to Strategic Patterns in a Robotic Soccer Domain

Andraž Bežek and Matjaž Gams
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
Andraz.bezek@ijs.si

Keywords: MAS modeling, strategy learning

Received: June 17, 2005

The paper presents an algorithm for multi-agent strategic modeling (MASM). The method applies domain knowledge and transforms sequences of basic multi-agent actions into a set of strategic action descriptions in the form of graph paths, agent actions, roles and corresponding rules. The rules, constructed by machine learning, enrich the graphical strategic patterns, which are presented in the form of graph paths. The method was evaluated on the RoboCup Soccer Server Internet League data. Tests showed that the constructed rules successfully captured some decisive offensive moves and some major defense flaws, although the description itself was a bit awkward and needed interpretation by a human expert.

Povzetek: Predstavljen je sistem, ki si uči strateških vzorcev obnašanja iz enostavnega opazovanja gibanja agentov v domeni robotskega nogometa.

1 Introduction

Multi-agent game modeling is related to the following task: How can external observation of multi-agent systems be used to analyze, model, and direct agent behavior? Analysis of such systems must capture complex world state representation and asynchronous agent activities. From pure numerical data researchers tend to construct complex knowledge-level structures, typically in the form of rules or decision trees. These high-level structures are useful when characterizing state space, but lack the ability to clearly represent temporal state changes occurred by agent actions. Comprehending simultaneous agent actions and complex changes of state space represents another problem. To capture such qualitative information, most often a graphical representation performs better in terms of human understanding.

There were two major goals in the research presented in this paper. One was designing strategic patterns from basic agent behavior, and the second one was to present the constructed knowledge in a graphical and symbolic form. This paper therefore addresses the problem of graphical and symbolic representation of strategic patterns, describes an algorithm capable of discovering strategic agent behavior, and enabling humans to understand and study the underlying behavioral principles.

The presented MASM algorithm translates multi-agent action sequence and observations of a dynamic, complex and multivariate world state into a graph-based and rule-based strategic representation. By using hierarchically ordered domain knowledge the algorithm is able to generate strategic descriptions and corresponding rules at different levels of abstraction. The MASM scheme is presented in Figure 1.

Our approach is applied on a RoboCup Simulated League domain (Noda et. al 1997, RoboCup 2004), a multi-agent domain where two teams of 11 agents play simulated soccer games. The domain accurately simulates a physical 2D soccer but introduces uncertainty by adding noise when calculating forces on objects. Continuous time is approximated with discrete cycles. All agents can move and act independently as long as they comply with soccer rules. Agents communicate with each other, but their visual and hearing perception is distance-limited. The domain is quite complex and represents a challenging multi-agent modeling task for computers, but its soccer-related content makes it comprehensible by humans familiar with soccer.

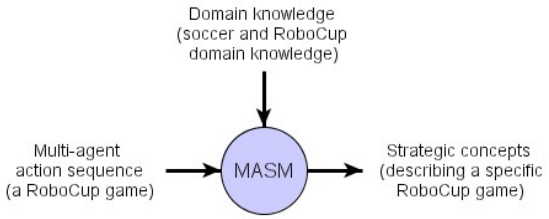


Figure 1: Multiagent System Modeling.

This paper is organized as follows. Section 2 thoroughly presents the MASM algorithm for creating graphical strategic paths. In Section 3, the learning algorithm is described for constructing symbolic strategic descriptions. An evaluation of the described method is presented in Section 4, and conclusion in Section 5.

2 Multi-Agent Strategic Modeling – the Graphical Part

Our multi-agent strategic modeling (MASM) algorithm transforms raw multi-agent action sequence into a set of discovered strategic action descriptions with corresponding rules. A strategic action description is a description of an agent behavior that exhibits some strategic activity. A strategic activity is a time-limited multi-agent activity that exhibits some important or unique domain-dependent characteristics.

Our approach is based on two basic processes:

1. Construction of graphical sequences of actions.
2. Learning symbolic rules.

The process in terms of creating increased higher-level structures is presented in Figure 2.

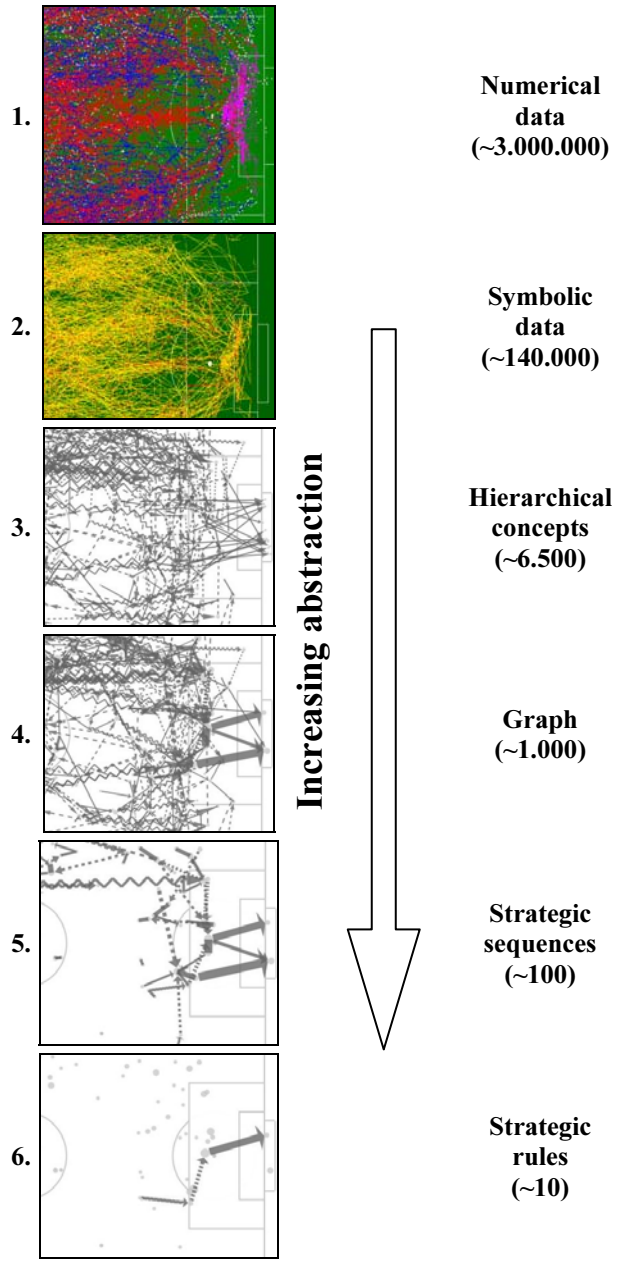


Figure 2: Construction of strategic patterns.

First 5 steps, presented in Figure 2, are going to be described in detail in this section and rule construction in the next section.

At the lowest level, RoboCup games are presented as time frames of agent and ball movements. For further information see (Cheny at al. 2003, RoboCup 2004).

	ball pos x	ball pos y	ball vel dx	ball vel dy	ball possessor l_team	ball possessor r_team	ball goal_distance l_team
min	-47.0280	-34.0000	-2.5906	-2.6121	0	0	14.4470
max	54.1653	34.0000	2.6116	2.6010	1	1	107.9837
type	double	double	double	double	char	char	double
4737	41.4884	-7.4706	0.7479	-0.4579	1	0	94.2848
4738	44.2138	-6.9890	2.5619	0.4527	0	0	96.9660
4739	46.6953	-6.4133	2.3326	0.5412	0	0	99.4024
4740	49.0566	-5.7881	2.2197	0.5877	0	0	101.7214
4741	51.1646	-5.2578	1.9816	0.4985	0	0	103.7979
4742	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4743	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4744	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4745	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4746	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4747	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4748	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4749	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4750	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4751	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4752	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4753	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4754	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4755	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4756	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4757	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4758	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4759	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4760	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4761	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4762	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4763	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4764	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355

Figure 3: Raw data in numbers.

In a RoboCup game there are approximately 6000 equidistant time frames and 512 attributes (together around 3,000,000 values of types integer, Boolean, and real). An example of data is presented in Figure 3.

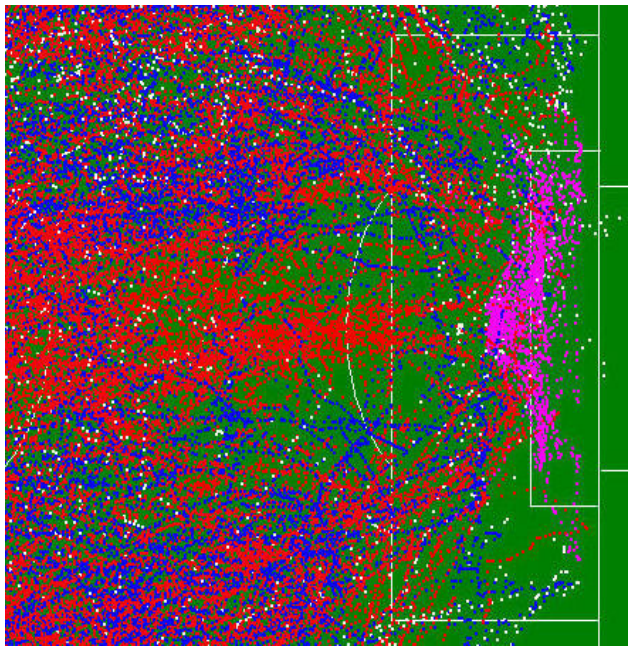


Figure 4: Visualization of raw data.

In Figure 4 there is a graphical presentation, i.e. visualization of the same data as presented in Figure 3. All game data was obtained from the Soccer Server Internet League (Robocup 2004).

In the next step, basic agent movement is transformed into basic agent actions using simple heuristic rules (Nair at al. 2002) such as: "An agent performs action "dash" if it increases speed." Each action lasts one time cycle. From a typical game, around 140.000 basic agent actions are obtained such as:

```
time player → action
--------------------
3192 LPlayer1 → catch
4012 LPlayer3 → kick
5400 RPlayer6 → dash
5900 RPlayer11 → turn
```

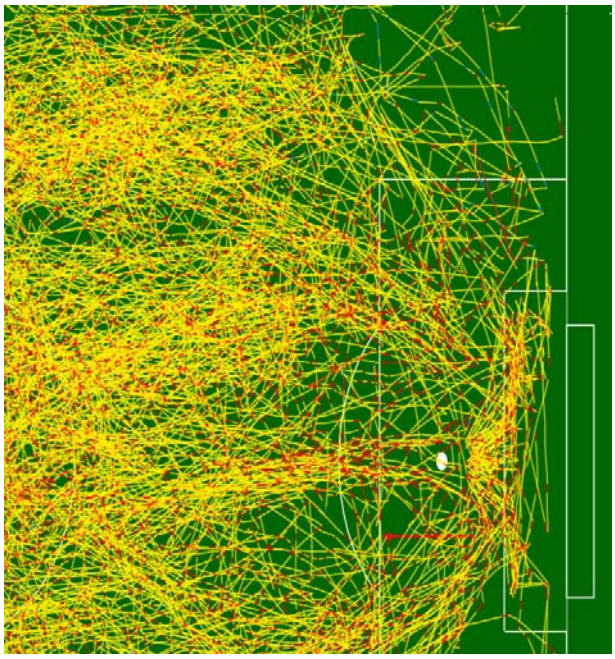


Figure 5: Basic agent actions.

In the next step, basic agent actions are transformed into higher-level actions using domain knowledge (Kaminka at al. 2002, Nair et al. 2003). The MASM algorithm exploits taxonomies, i.e. hierarchical representations of domain concepts. A concept x in a taxonomy is an ancestor of a concept y, $x \leftarrow y$, if it exhibits more general concept than the concept y. The rationale behind using hierarchically ordered domain knowledge is that this allows the MASM algorithm to travel up and down in the hierarchy to produce more or less abstract descriptions. Specifically, the MASM algorithm makes use of:

- a taxonomy of agent roles
- a taxonomy of agent actions
- taxonomies of binary domain features.

Taxonomies were created from the Internet, using Dictionary Of Soccer Terms, Concepts & Rules. Parts of these taxonomies are presented in Figure 6, determining agent roles and actions.

As agents in MAS can change roles and thus change their behavior (Nair et al. 2003), agent roles are assigned dynamically during agent activity. Each agent action is assigned with its corresponding hierarchical representation. Domain features are used to identify the truth of some particular domain feature but only with an association with another agent. For example, in a RoboCup domain the feature *HasBall* is true only for agent which controls the ball, and is false for all other agents. Agent's roles and actions are used to describe the activity of agents, while domain features are used to describe the domain state space. An example assignment of soccer role and action concepts is presented in Figure 6, some examples are presented below:

```
time player_role → action, action_duration
-------------------------------------------
3192 LTeam.Goalkeeper → catch, 1
4012 LTeam. LeftForward → pass_to_player, 10
5400 RTeamRightFullback → speed_dribble, 12
5900 RTeam.LeftMidfielder → intercept, 8
```

Around 1000 such high-level agent actions are constructed for a typical game.

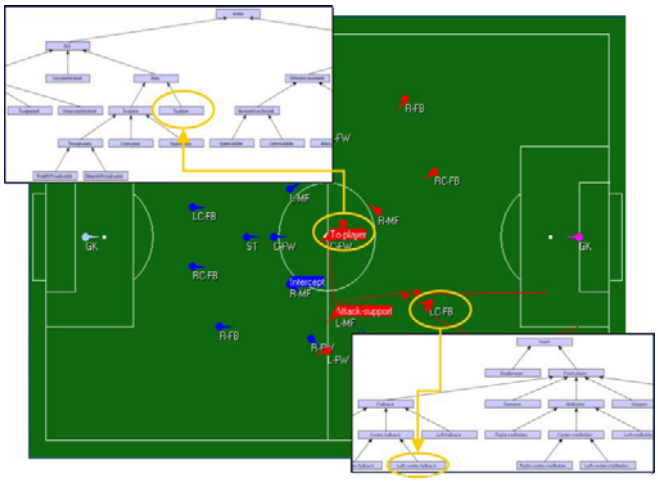


Figure 6: High-level agent actions.

After agent-role assignment, an *action graph* (AG) is constructed with the goal to create action patterns in a graphical form of paths. An action graph is a directed graph, where nodes represent state space at the start of agent action and connections correspond to agent actions. Nodes a and b are connected, $a \rightarrow b$, if an action, represented by node a, is followed by an action, represented by node b. Terminal actions (i.e. the last action in an action sequence) are connected to a terminal node. For example, an action sequence {a,b,c} is represented as an AG: $a \rightarrow b \rightarrow c \rightarrow c_{end}$. Node positions are calculated from agent positions in a domain space. An appropriate hierarchical action and role concepts are assigned to each node (Bezek 2005). This enables the MASM algorithm to generate more abstract descriptions of agent role and actions. Each node also keeps an

original action instance (i.e. time cycle of an action in a soccer game) of the represented action. A more detailed description of action graphs and the construction process is given in (Bezek 2004). An example action graph, obtained from actions in a RoboCup game, is presented in Figure 7.

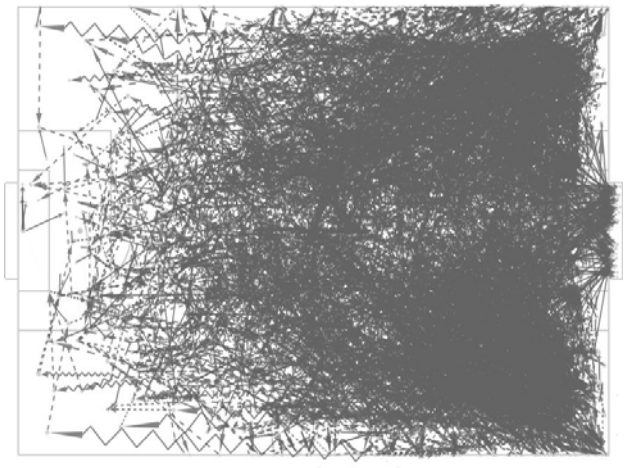


Figure 7: An action graph.

Complex action graphs with many nodes (see Figure 7) are difficult to present in a transparent manner and are thus difficult to comprehend by humans. A reasonable approach to overcome this problem is to reduce the number of nodes while at the same time preserving attained action concepts. This can be accomplished with hierarchical clustering of graph nodes. By merging the nearest two graph nodes, we generate a new node that represents common role and action concepts (i.e. first hierarchically common parent of both concepts). The rationale behind the merge process is that actions, frequently occurring near in a domain space, define strategic concepts. The distance between graph nodes is defined as a weighted sum of distances between node positions and conceptual distances between role and action concepts. The merging process is then iterated. A more in-depth description of the distance function and the whole abstraction process is described in (Bezek, Gams 2005).

The clustering process results in an *abstract action graph* (AAG), which is an action graph where graph nodes represent more than one agent action. It is expected that abstract action graph describes agent behavior in a more abstract way than the original action graph. An abstract action graph, where minimal distance between nodes is greater than *dist*, is labeled AAG_{dist}. Such graph can be achieved with repeated merging of nearest nodes until the minimal distance between nodes is grater than *dist*. An *action description* of a node in AAG_{dist} is a combination of a node position, corresponding action and role concepts, and a parameter *dist*. AAGs with greater value *dist* represent actions in a more abstract way that AAGs with a lower *dist* value. Therefore, the value of a *dist* parameter can be regarded as a value of abstraction of an AAG. An example of an abstract action graph A_{10} is presented in Figure 8.

In Figure 8 there are several connected arrows of different length, positions and thickness. One example of transformation from single actions into an aggregated one is shown in Figure 9. It represents a common and successful attack on the right side of the field, resulting in a successful shoot on a goal. It is an example of a desired graphical representation of a strategic pattern.

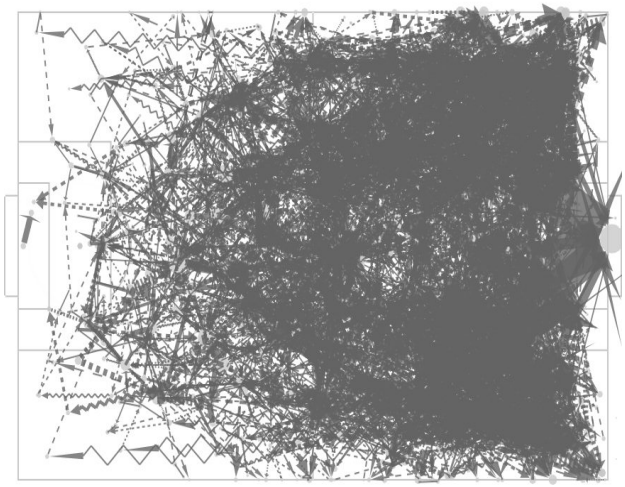


Figure 8: An abstract action graph (AAG_{10}).

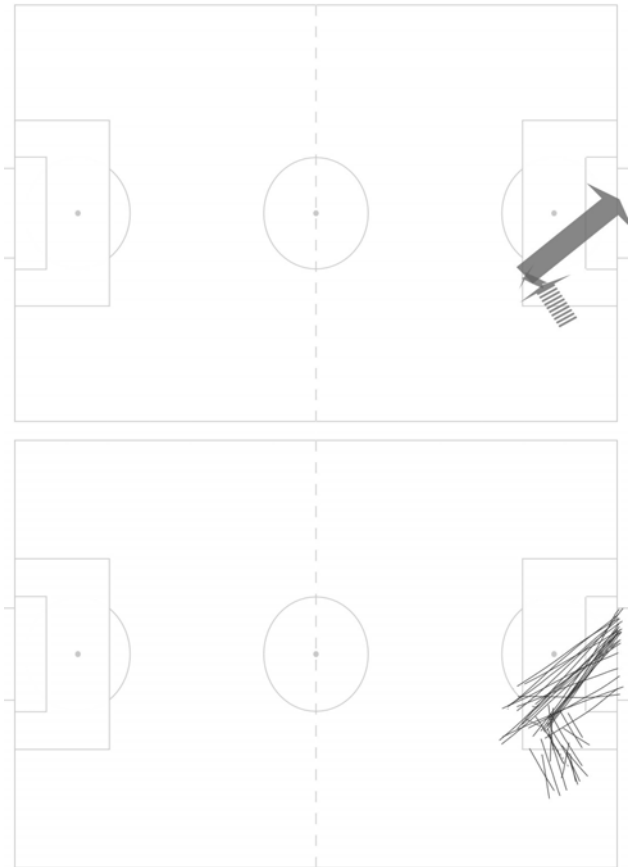


Figure 9: Transformation of agent actions into abstract action sequence as part of action graph (AAG).

This strategic abstraction of agent actions is based on clustering (Hirano et al. 2004, Riley at al. 2001), and on conceptual distance, based on the domain taxonomies. As a result, around 1000 of such structures are generated from a typical game:

```
role → action: {(action_start, duration)+ }
-------------------------------------
LTeam.Goalkeeper → catch:
{(412, 1), (501, 1), (3192,1)}
LTeam.Forward → pass:
{(1412, 5), (3401,12), (4012,10) , (5573,7)}
RTeam.Defender → speed_dribble:
{(1607,16), (2372,9), (5400,12), (5521,22)}
RTeam.Midfielder → intercept:
{(392, 4), (4509, 9), (5900, 8)}
```

A list of subsequent actions with corresponding symbolic description represents a *strategy,* i.e. a similar and frequent multi-agent activity that leads to a strategic situation. In an abstract action graph it is represented as a path. Path nodes thus represent a sequence of strategic actions.

What remains is construction of rules, as indicated by step 6 in Figure 2.

3 Construction of Rules

First it should be noticed that strategies vary in several parameters, such as number of actions (typical 2 to 4), abstractness of actions (corresponding to the number of single actions aggregated into one abstracted action), location, direction etc. In general, a strategy generated from AAG with a greater *dist* value is more abstract that the one generated from AAG with lower *dist* value. The strategy in Figure 9 was created using level of abstractness 8 (=*dist*). The strategic action sequence is presented in Table 1. From Table 1 and Figure 9 the strategic action sequence can be described as (no. of positive examples in parentheses):

Forward player **passes** a ball to a teammate (21 +),
who successfully **dribbles** (10 +), and
shoots towards a goal (23 +).
As a result, the ball ends in a goal (23 +)."

LTeam.FW: Pass-to-player	LTeam.FW: Control-dribble	LTeam.FW: Successful-shoot	LTeam.Field-player: Successful-shoot-(end)

Table 1: A strategic action sequence.

The action sequence in Table 1 is graphically presented as the path consisting of three connected arrows in Figures 9 and 10. Each action (an arrow) graphically starts from a circle (Figure 10) which corresponds to the neighborhood including aggregated actions. All circles in Figure 10 correspond to all aggregating neighborhoods.

As each node/circle defines a unique action concept it can be used to generate rules that describe this specific agent action. In particular, we generate data for rule inducing algorithms as follows: Positive examples are action instances in a target node and negative examples as instances in nearby nodes (i.e. near misses). For each instance we generate all pairs of agent role-domain feature and store the true ones.

We tested several approaches with association rules (Agrawal et al. 1994, Srikant et al. 1995), but due to complex representations we found the standard feature-value approach as not suitable. Namely, agents dynamically change roles and thus it is very difficult to generate feature values for all roles. Therefore, instead of feature-values we applied set-valued attributes that are attributes whose domains are sets instead of single values. In this way, each feature corresponds to one set-valued attribute where the value is a set of agent roles, whose corresponding agent-feature pair is true.

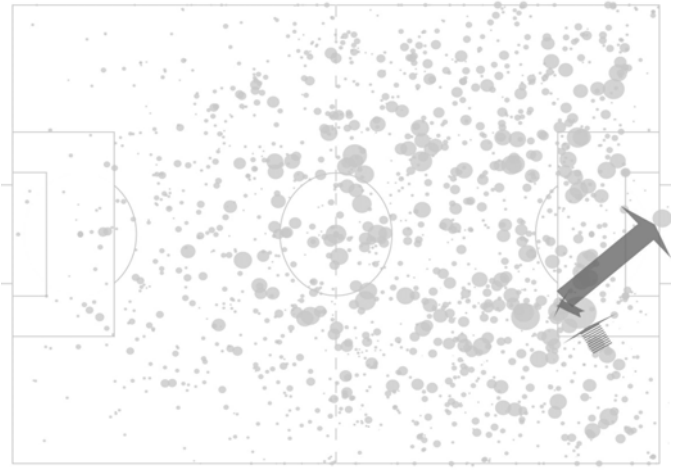


Figure 10: Strategy as a path in the abstract action graph, and all potential learning examples as circles.

By using set-valued rule inducer, such as SLIPPER (Cohen et al. 1999), the MASM algorithm is able to generate rules that describe actions in a strategy. In a typical experiment, 10 games of the same team were taken as input, and SLIPPER was applied on each node in a strategic path.

For example, a rule describing a node, which represents an action concept "Successful-shoot" performed by an agent with a role "left team center midfielder", is presented in Table 2.

ball:Penalty-box ∧ ball:Right-half ∧ ball:Fast ∧ LTeam.C-MF:Has-ball ∧ LTeam.R-FW:Moving-away ∧ LTeam.R-FW:Medium-dist ∧ RTeam.R-FB:Back ∧ RTeam.C-FB:Back ∧ RTeam.L-FB:Back.

Table 2: Symbolic description of a successful shoot by a left team's center midfielder.

There are several parameters that influence the learning algorithm, and the influence of distance is indicated in the following examples. The distance parameter corresponds to the number of learning examples. Since it seems reasonable to include all

positive examples, because there are typically only around 10 or 20 of them (note that these are strategic patterns that actually occur in a game), the parameter varies the number of negative examples.

- All negative examples:

LTeam.FW:Pass-to-player (#+21 #-6987) <=
(LTeam.R-FW:Has-ball = 1) AND (LTeam.L-FB:Incoming-slow = 1) AND (RTeam.GK:Incoming = 1)
(*there are 21 positive examples and 6987 negative*)
LTeam.FW:Control-dribble (#+10 #-6998) <=
(LTeam.R-MF:Near = 1) AND (LTeam.L-MF:Attacking-third = 1) AND (LTeam.C-FW:Center-of-the-field = 1)
LTeam.FW:Successful-shoot (#+23 #-6985) <=
(LTeam.LC-FB:Moving-away-slow = 1) AND (RTeam.R-MF:Attacking-third = 1) AND (LTeam.R-MF:Fast = 1) AND (RTeam.R-FB:Far = 1) AND (RTeam.GK:Faster = 1) AND (RTeam.L-FW:Moving-away = 1) AND (LTeam.C-FW:Medium-distance = 1) AND (RTeam.C-MF:Fast = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-6787) <=
(Ball:Opponent-goal = 1)

- Only negative examples with distance <= 16

LTeam.FW: Pass-to-player (#+21 #-854) <=
(LTeam.C-FW:Incoming-fast = 1) (* no. of negative examples here is 845*)
LTeam.FW:Control-dribble (#+10 #-1425) <=
(LTeam.R-MF:Near = 1)
LTeam.FW:Successful-shoot (#+23 #-1447) <=
(LTeam.R-FB:Moving-away = 1) AND (LTeam.R-FW:Moving-away = 1) AND (RTeam.R-FB:Far = 1) AND (LTeam.R-MF:Attacking-third = 1) AND (RTeam.GK:Incoming = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-213) <=
(RTeam.GK:Right-half = 1) AND (RTeam.GK:Back = 1) AND (LTeam.L-MF:Center-of-the-field = 1) AND (LTeam.L-FW:Medium-distance = 1)

- Only negative examples with distance <= 8

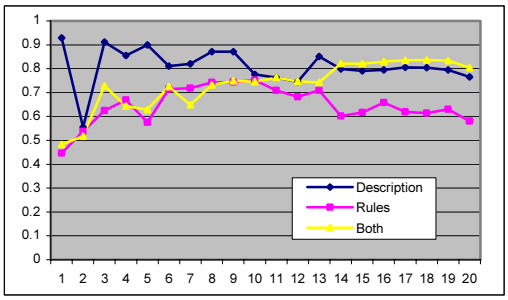
LTeam.FW: Pass-to-player (#+21 #-265) <=
(LTeam.C-FW:Incoming-fast = 1) (* no. of negative examples here is 845*)
LTeam.FW:Control-dribble (#+10 #-513) <=
(LTeam.RC-FB:Fast = 1)
LTeam.FW:Successful-shoot (#+23 #-573) <=
(RTeam.L-FW:Moving-away = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-113) <=
(RTeam.GK:Right-half = 1) AND (LTeam.LC-FB:Right = 1)

- Only negative examples with distance <= 4

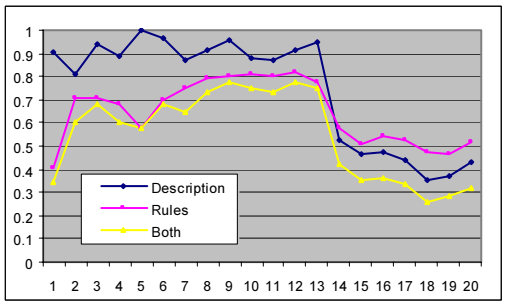
LTeam.FW: Pass-to-player (#+21 #-105) <=
(RTeam.R-FB:Incoming-fast = 1) (* no. of negative examples here is 105*)
LTeam.FW:Control-dribble (#+10 #-239) <=
(RTeam.GK:Right = 1)
LTeam.FW:Successful-shoot (#+23 #-200) <=
(LTeam.R-FB:Moving-away-slow = 1) AND (LTeam.R-FW:Moving-away = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-112) <=
(LTeam.R-MF:Right-wing = 1) AND (RTeam.GK:Right = 1) AND (LTeam.L-MF:Left-half = 1) AND (RTeam.GK:Short-distance = 1)

4 Measurements

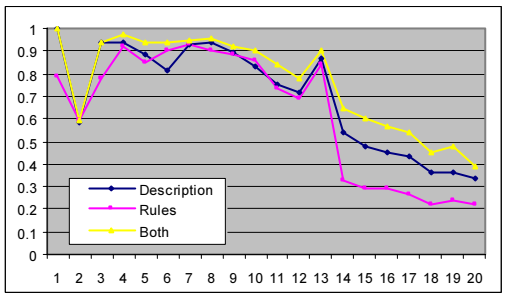
We evaluated the MASM approach on 10 RoboCup games played during SSIL (Robocup 2004). A leave-one-out strategy was used to generate 10 learning tasks. A pre-determined strategy, shown in Table 1 and in Figure 10, was used as a reference and was generated on all 10 games, for AAG_1 to AAG_{20}. For each learning task, a strategy was generated on 9 games and tested on the remaining game, again for AAG_1 to AAG_{20}. Tests measured the quality of action descriptions, the quality of an average rule and the quality of joint use of rules and action descriptions. Figure 11 presents averaged results obtained during 10 tests where x-axis presents the value of a parameter *dist*. These results indicate that a) the accuracy of action descriptions is approximately constant regarding abstraction. However, the accuracy of rules increases until it peaks at dist=10 and then slowly decreases. This is expected because for lower abstractions, nodes represent only a few action instances consequently prohibiting rule inducer to generate good rules.



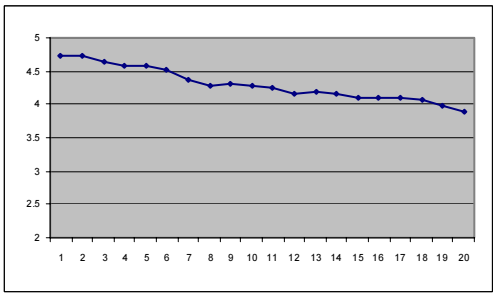
a) Accuracy



b) TP rate



c) Precision



d) Abstractness

Figure 11: Accuracy a), true-positive rate b) and precision c) measured in relation to the abstractness level presented on x-axis. Abstractness of attributes is in d).

With high *dist* values, nodes represent different action concepts, thus producing more abstract and less accurate rules. But using rules and action descriptions together gives the best results with higher dist values.

When measuring true-positive rate, i.e. the percentage of correctly classified true cases, all test scenarios give similar results as shown in Figure 11 b): the quality of classifying true cases increases until about dist=12, and then quickly drops. The similar phenomenon is observed when measuring precision, that is the rate of correctly classifying the true cases, shown in Figure 11, c). This can be explained by generating too abstract strategies that represent the agent behavior in a too abstract way.

The last test was performed to verify if the abstraction process generates more abstract descriptions. For this test we measured the abstraction of generated rules, defined as an average feature depth in the feature taxonomy for features used in rules. The results, presented in Figure 11 d), clearly show that the average feature depth is negatively correlated with the parameter *dist*. This proves that as *dist* value increases, the rules contain more abstract features.

The constructed strategic patterns were also examined by the research team and a human expert. We studied the games on the screen in real time and the constructed strategic patterns. Firstly, we realized that the direct computer output was unintelligible for a non-computer specialist. Second, the constructed computer output had to be studied also by the research team since quite often the meaning of constructed features had to be figured out. For example, instead of a meaningful "fast ball" the actually constructed feature was "distance between a ball and a player is growing fast". Another annoying property of the learning algorithm was that sometimes quite irrelevant features were constructed, at least from the point of human understanding. But in our joint overall opinion, the algorithm finds some significant features (moves), which is quite a success since the algorithm has no knowledge whatsoever about rules of soccer or any predefined knowledge about strategies, i.e. a list of potential soccer strategies.

5 Conclusion

We have designed and implemented the MASM algorithm as a general domain-independent framework for discovering strategic behavior of multi-agent systems. The only domain-specific knowledge was introduced in the form of role, action and domain feature taxonomies. We assume that changing a domain should be a straightforward task that would require changing specific domain-knowledge in a similar form. We believe that there is a wide range of possible domains that can be exploited by the MASM since its essence is a stepwise abstraction in the domain-space.

The tests show that the system with 30.000 source code lines achieves reasonable results in terms of accuracy, true-positive rate and precision. Our tests also confirm that the increased abstraction process generates more abstract descriptions of agent activities.

However, there are some open questions that need to be addressed. First, the MASM system was evaluated only on the RoboCup domain with a limited number of tests. Although authors believe that no major problem should emerge when introducing another domain, this should be verified in practice. Second, while the output of the MASM system seemed promising to the research team and the soccer coach performing preliminary evaluation, this should be systematically verified by a number of unrelated humans and soccer experts. The third open question is how to objectively specify relevant strategic situations.

Overall, the MASM algorithm was able to create human comprehendible strategic descriptions in the form of graphical arrows and related strategic rules with reasonable accuracy from basic agent observations in a RoboCup games. This seems quite promising since the system had only limited domain knowledge.

References

[1] R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules. Proc. 20th Int. Conf. Very Large Data Bases, VLDB, 1994.

[2] A. Bezek: Modeling Multiagent Games Using Action Graphs. Proceedings of Modeling Other Agents from Observations (MOO 2004), New York, 2004.

[3] A. Bezek: Discovering Strategic Multi-Agent Behavior in a Robotic Soccer Domain, Proceedings of AAMAS 05, Utrecht, 2005.

[4] A. Bezek, and M. Gams: Discovering strategic multi-agent behavior in a robotic soccer domain. Proceedings of Information Society 2005, Ljubljana, 2005

[5] M. Cheny et al.: Users Manual for RoboCup Soccer Server, 2003.

[6] W. W. Cohen and Y. Singer: A simple, fast, and effective rule learner. Proceedings of the sixteenth national conference on Artificial intelligence, pp.: 335 - 342, Orlando, United States, 1999.

[7] Dictionary Of Soccer Terms, Concepts & Rules, http://www.soccerhelp.com/Soccer_Tips_Dictionary_Terms.shtml.

[8] S. Hirano and S. Tsumoto: Finding Interesting Pass Patterns from Soccer Game Records. The Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2004).

[9] I. Noda et. al: Overview of RoboCup-97. In Hiroaki Kitano, editor, RoboCup-97: Robot Soccer World Cup I, pp. 20-41. Springer Verlag, 1997.

[10] G. Kaminka, M. Fidanboylu, A. Chang, and M. Veloso: Learning the sequential coordinated behavior of teams from observations. Proceedings of the RoboCup-2002 Symposium, June, 2002.

[11] R. Nair, M. Tambe, S. Marsella and R. Raines: Automated assistants for analyzing team team behaviors Journal of Autonomous Agents and Multiagent Systems. JAAMAS, 2002.

[12] R. Nair, M. Tambe, and S. Marsella: Role allocation and reallocation in multiagent teams: Towards a practical analysis. Proceedings of the second International Joint conference on agents and multiagent systems (AAMAS), 2003.

[13] P. Riley and M. Veloso: Coaching a Simulated Soccer Team by Opponent Model Recognition. Proceedings of the Fifth International Conference on Autonomous Agents, 2001.

[14] RoboCup 2004: RoboCup Simulation League. RoboCup04 world cup game repository, http://carol.science.uva.nl/~jellekok/robocup/rc04/, 2004.

[15] R. Srikant and R. Agrawal: Mining Generalized Association Rules. Future Generation Computer Systems, 1995.