

Multi-Agent Strategic Modeling in a Robotic Soccer Domain

Andraz Bezek, Matjaz Gams

Department of Intelligent Systems

Jožef Stefan Institute

{andraz.bezek, matjaz.gams}@ijs.si

Ivan Bratko

Faculty of Computer and Information Science

University of Ljubljana

bratko@fri.uni-lj.si

ABSTRACT

This paper presents an algorithm for multi-agent strategic modeling (MASM) applied in a robotic soccer domain. It transforms a multi-agent action sequence into a set of strategic action descriptions in a graphical and symbolic form. By using hierarchically ordered domain knowledge, the algorithm is able to generate graphic and symbolic strategic action descriptions together with corresponding rules at different levels of abstraction. The method was evaluated on the RoboCup Soccer Server Internet League data.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning - *Concept learning*

General Terms

Algorithms, Experimentation.

Keywords

Multi-agent strategic modeling, strategy, RoboCup.

1. INTRODUCTION

Multi-agent modeling deals with the following task[15]: How can external observations of multi-agent systems be used to analyze, model, and direct agent behavior? Analysis of such systems must thus capture complex world state representation and asynchronous agent activities. Instead of studying pure numerical data researchers often use knowledge-level structures, such as rules and decision trees. Although being extremely useful when characterizing state space, they lack the ability to clearly represent temporal state changes occurred by agent actions. To comprehend simultaneous agent actions and complex changes of state space,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00.

most often a combination of graphical and symbolical representation performs better in terms of human understanding.

Multi-agent strategic modeling (MASM) is a process that is able to discover a strategic activity from raw multi-agent action data. A strategic activity is a time-limited multi-agent activity that exhibits some important or unique domain-dependant characteristics.

This paper addresses the problem of graphically and symbolically representing strategies of multi-agent systems (MAS). Our goal was to create an algorithm that would discover strategic agent behavior and enable humans to understand and study the underlying behavioral principles. The task is to translate multi-agent action sequence and observations of a noisy, dynamic, complex and multivariate world state into a graph-based and rule-based representation. By using hierarchically ordered domain knowledge the algorithm should be able to generate strategic action descriptions and corresponding rules at different levels of abstraction. More abstract descriptions also tend to be more noise resistant.

Our approach is applied in a Multi-Agent Strategy Discovering Algorithm (MASDA) and tested on a RoboCup Simulated League domain (RoboCup) [12], a multi-agent domain where two teams of 11 agents play simulated soccer games. The physical 2D soccer is simulated with additional 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 can freely communicate with each other, but as in real world their visual and hearing perception is distance-limited. The domain is at the same time a complex MAS and easily understandable by humans due to its soccer-related content.

An example of a sequence of RoboCup actions is presented in Figure 1. The four frames show a successful direct-attack action. The players are presented as filled circles with a pointer indicating body direction. The attacking team is black while the defending team is gray. The ball is a white circle with black border; its trace is drawn throughout the sequence. Lines represent traces of movement. The action sequence in Figure 1 is typically described by humans as: "The left team's left forward player dribbles from the left half of the middle third into the penalty box. He then passes a ball to the center forward in the center of the penalty box, where the receiver successfully shoots into the right part of the goal." This sequence is graphically presented in Figure 2 consisting of **player roles**, **player actions** and factual **domain-related descriptions**, such as positions of involved players.

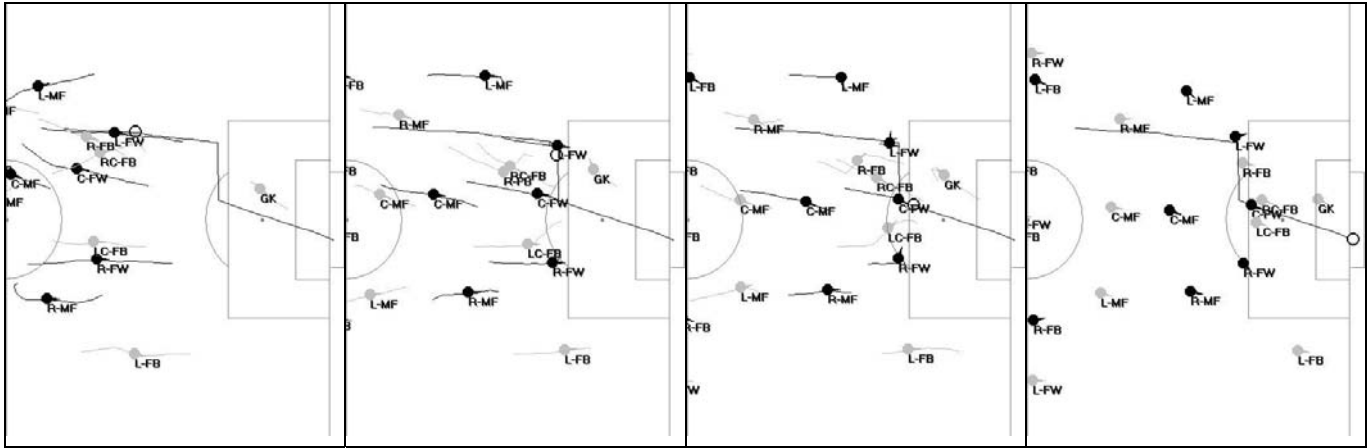
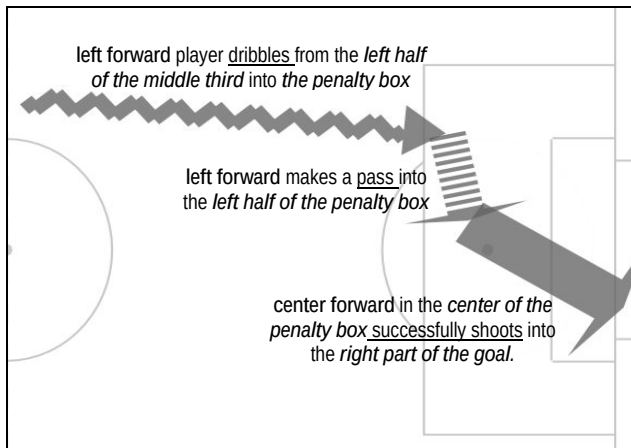


Figure 1: Four frames from a RoboCup game showing a successful direct-attack action sequence.

The diagrammatic action representation is commonly used to help understanding spatial and temporal activity, as shown in Figure 2. Since the aim of MASD is to discover strategic activity and to describe this activity in a human-understandable form, MASDA uses a similar representation, with a diagram, action sequence and rules that describe factual domain features.



**Figure 2: An example of a soccer strategy:
a left side direct attack.**

This paper is organized as follows. Section 2 reviews the related work. Section 3 presents MASDA. Section 4 describes an example of a generated strategy. In Section 5 an evaluation of the described method is given, followed by discussion in Section 6.

2. RELATED WORK

The task of graphically representing agent actions has been addressed by several multi-agent graphical models, most notably by multi-agent influence diagrams [8] and influence diagram networks [15]. However, all these are concerned with agent decision making, and not agent modeling, and are therefore not suitable for representing observed strategic agent behavior.

The work [11] from Intille deals with the related task of recognizing multi-agent actions in the American football domain. However, the focus in that work is on recognition, using a database of plans, knowledge about football, computation of perceptual features from the trajectories, and propagation of uncertain reasoning, the recognition system developed can determine if the example of a game play is in fact an example of a "p51curl" play. The drawback with their approach is that they explicitly make use of a database of plans and cannot discover new behavior. The paper [5] from Hirano and Tsumoto presents a novel method for finding interesting pass patterns from soccer game records. Taking two features of the pass sequence – temporal irregularity and requirements for multiscale observation – into account, they have developed a comparison method of the sequences based on multiscale matching. The method is used with hierarchical clustering to produce clusters of spatial pass patterns. Experimental results soccer game records demonstrate that the method is able to discover some interesting pass patterns that may be associated with successful goals. This work is concerned primarily with spatial pass patterns while out process also captures important domain-specific knowledge.

There has been extensive research about quantitative modeling in the RoboCup domain. Unfortunately, the problem of constructing human-readable representations of multi-agent activity has received little interest. The work from Raines et al. [13] focused on an automated assistant for analyzing agent teams. Their system ISAAC performs post-hoc, off-line analysis of teams using agent-behavior traces in the domain. The analysis is performed bottom-up, using data mining and inductive learning techniques. ISAAC is able to discover interesting patterns based on user provided features, thus combining the analytical capabilities of the machine with the domain knowledge of the user. Their system is similar to ours in terms of domain knowledge and rule induction. However, our approach explicitly exploits abstraction and provides graphic representation for detected strategies. Work from Kuhlmann et al. [9] deals with the problem of formulating advice-giving as a learning problem. They are able to learn and predict agent behavior from past observations and automatically generate advice to improve its team performance. Although their work concentrates on advice giving task, they successfully construct rules that predict the next significant action events. Our work is

similar in terms of rule induction; however we focus towards increased human comprehension and visual representation as opposed to purely classification accuracy. Work from Kaminka et al. [7] addresses the problem of unsupervised autonomous learning of the sequential behaviors of agents. They translate observations of the complex world state into a time-series of recognized atomic behaviors. The time series are then analyzed to find repeating or statistically significant subsequences characterizing each team. Their work addresses only frequent action subsequence matching while we construct graph-based action model that provides spatial, sequential, and temporal information about agent actions. Kaminka et al. [6] later introduced a behavior-based approach to plan recognition, in which hierarchical behaviors are used to model the observed robots. Efficient symbolic algorithms for answering key queries about observed robots are also presented. However, their approach explicitly utilizes a hierarchical behavior library while our process creates graph-based action models on-the-fly. The work from Sukthankar and Sycara [16] presents a cost minimization approach to the problem of human behavior recognition in a military domain. Using full-body motion-capture data acquired from human subjects, the system is able to recognize the behaviors that a human subject is performing. Low-level motion classification is performed using support vector machines (SVMs) and a hidden Markov Model (HMM); output from the classifier is used as an input feature for the behavior recognizer. Given the dynamic and highly reactive nature of the domain, their system is able to handle behavior sequences that are frequently interrupted and often interleaved. While their approach can identify different behaviors using behavior library our approach detects strategic activity without prior strategic knowledge.

3. THE ALGORITHM

In this section we present MASDA, which is capable of discovering strategic multi-agent activity and describe it graphically and symbolically. The algorithm can be presented as a three-step process: a preprocessing step, a graphical-description generation step and a symbolic-description generation step. Each step is further split into smaller tasks, as presented here, with data structures in parenthesis:

- Step 1: Data preprocessing:
 - Detection of actions in raw data (attributes)
 - Action sequence generation (symbolic sequence)
 - Introduction of domain knowledge (hierarchical concepts)
- Step 2: Graphical description creating:
 - Action graph creation (action graph)
 - Abstraction process (abstract action graph)
 - Strategy selection (graph path)
- Step 3: Symbolic description learning:
 - Action description generation (action description)
 - Rule data generation (set-valued attributes)
 - Rule induction (rules)

The output from each step is the input for the next one. In the first two tasks of the first step MASDA detects actions in raw multi-agent data and generates corresponding action sequences. Detailed description of preprocessing [1] is omitted due to lack of space. The domain knowledge is introduced (task 3 of step 1) in Section 3.1. The process of generating graphical descriptions

(step 2) is described in Section 3.2. In Section 3.3 we describe the process of symbolic descriptions generation (step 3).

3.1. Introduction of Domain Knowledge

MASDA exploits domain knowledge through taxonomies of domain concepts. *Taxonomy* T is a hierarchical representation of domain concepts. A concept $x \in T$ is an ancestor of a concept $y \in T$, $x \leftarrow y$, if it denotes a more general concept than the concept y . Hierarchically ordered domain knowledge allows MASDA to travel up and down in the hierarchy to produce more or less abstract descriptions. Specifically, MASDA makes use of:

- a taxonomy of agent roles (T_{roles}), shown in Figure 4
- a taxonomy of agent actions ($T_{actions}$)
- taxonomies of binary domain features ($T_{features}$)

Given a hierarchically ordered structure, we can define the *depth of a concept*. It is the length of the shortest path from the root to the given concept. Consequently, the smaller the depth is, the more abstract the concept is. The root concept has a depth 0.

As agents in MAS can change roles and thus change their behavior [10], agent roles are assigned dynamically during agent activity. Similarly, each detected agent action is assigned its corresponding hierarchical representation. While agent roles and actions are used to describe the activity of agents the domain features are used to describe the truth of some particular domain feature in a domain state space. In our case, domain features are only defined for moving entities - the ball or an agent. They are used to describe agent positions, their relation to the ball and their states. For example, in a RoboCup domain, the feature *HasBall* is true only for the agent that controls the ball (E.g.: *agent1:HasBall=true*) and is false for all other agents (E.g.: *agent2:HasBall=false*). A part of ball-related domain feature taxonomy is shown in Figure 3.

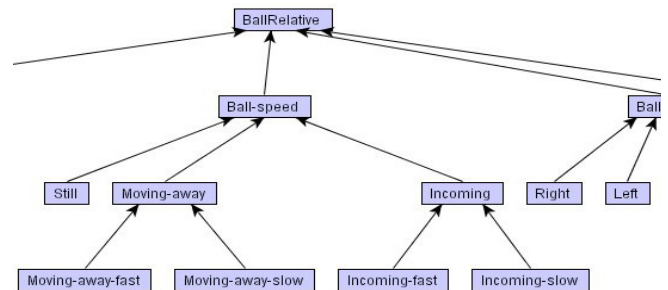


Figure 3: A part of ball-related soccer domain features.

We defined the following three domain taxonomies:

- Position-related taxonomy defines concepts describing whether the ball or an agent is in certain spatial region of the playing field. For example: *left-wing*, *center-of-the-field*, *right-wing*, *left-half*, *right-half*, *defending-third*, *defending-third-danger-zone*, *defending-third-penalty-box*, *defending-third-goal-box*, *middle-third*, *center-circle*, *attacking-third*, *attacking-third-danger-zone*, etc.
- Speed-related taxonomy defines the speed of an agent and ball: *stationary*, *moving*, *fast*, *medium*, and *slow*.

- Ball-related taxonomy defines relations of players regarding the ball, as shown in Figure 3. For example: *has-ball*, *left*, *right*, *front*, *back*, *incoming*, *moving-away*, *incoming-fast*, *incoming-slow*, *moving-away-fast*, *moving-away-slow*, *still*, *immediate*, *near*, *far*, etc.

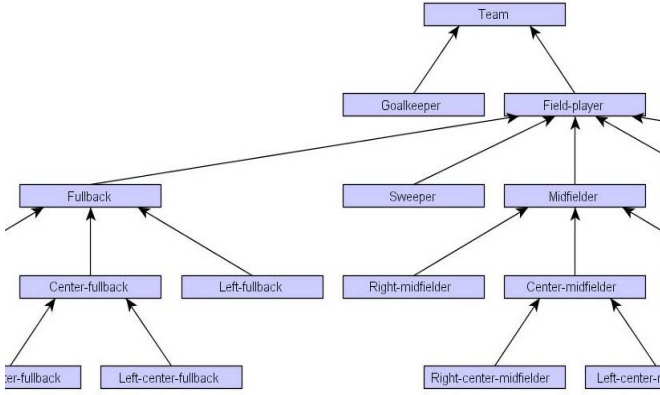


Figure 4: A part of role concept taxonomy.

3.2. Graphical representation

In this section, we describe the process of constructing graphical action representation from the observed action sequence that was generated in the preprocessing step.

3.2.1 Action Graph Representation

An *action graph* (AG) is a directed graph, where connections correspond to actions and nodes represent state space at the beginning of each action. Terminal actions (i.e. the last actions in action sequences) are represented as connections to terminal nodes, which are nodes with no outgoing connections and define the state space at the end of action. For example, an action sequence (α, β, γ) is represented as an AG:

$$a \xrightarrow{\alpha} b \xrightarrow{\beta} c \xrightarrow{\gamma} c_{terminal}$$

Node positions correspond to agent positions in a domain space. Each node is also assigned a common action- and role-parent (see Table 1) of outgoing actions from action and role taxonomies. This enables MASDA to generate abstract descriptions of agent role and actions. Each connection also holds start times and durations of the represented actions.

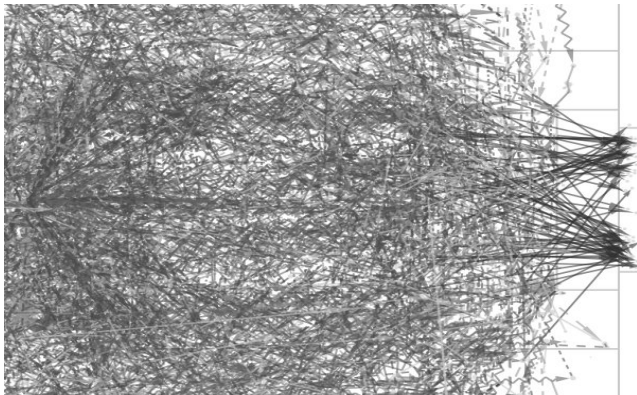


Figure 5: An example action graph.

A more detailed description of action graphs and the construction process is given in [1]. An example action graph, obtained from actions in 10 RoboCup games, is presented in Figure 5.

3.2.2 The Abstraction Process

Complex action graphs with many nodes are difficult to visualize, and are thus difficult to comprehend by humans. An approach to overcome this problem is to reduce the number of nodes. However, one would like to reduce action graph nodes in a manner consistent with captured 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. the most specific common ancestor of both concepts). The rationale behind the merge process is that actions, frequently occurring near in domain space, define some important strategic concepts. The definition of 'near' thus solely governs how the abstraction process combines action descriptions. Although the distance function is strictly domain-dependent it must utilize conceptual distance between actions, agent roles, and domain positions. Formally, the distance between nodes a and b is a function $distance(a, b)$, which must comply with the following three distance axioms [3]:

1. Non-negative definiteness: $\forall a, b:$
 $dist(a, b) \geq 0 \wedge dist(a, b) = 0$ iff $a = b$
2. Symmetry: $dist(a, b) = dist(b, a)$
3. Triangular inequality:
 $\forall a, b, c: dist(a, b) \leq dist(a, c) + dist(c, b)$

We defined the distance between graph nodes as a weighted sum of distances between node positions and graph distances between role and action concepts. The weights were assigned by trial and error to produce meaningful results. A more in depth description of the distance function and the whole abstraction process is described in [2].

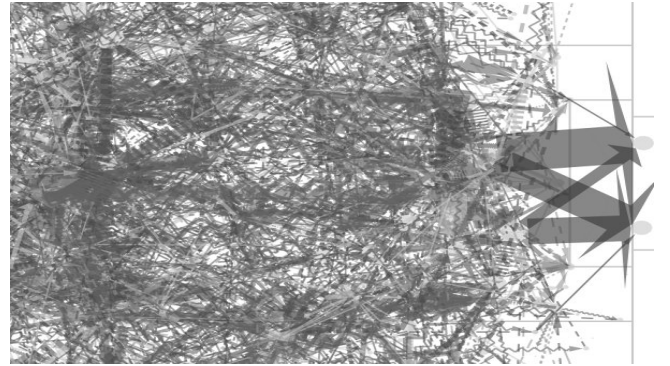


Figure 6: An abstract action graph (AAG_8).

The clustering process results in an *abstract action graph* (AAG), which is an action graph where graph nodes represent more than one agent action. 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 or equal than some value $dist$, is labeled AAG_{dist} . Such graph can be created with repeated merging of nearest nodes until the minimal distance between nodes is greater 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 than AAGs with a lower $dist$ value. Therefore, the value of parameter $dist$ can be regarded as the degree of abstraction of an AAG. An example abstract action graph AAG_8 is presented in Figure 6. The abstraction process can be stated in a pseudo-code as follows:

Input:

- action graph: G
- node distance function: $distance(n_1, n_2)$
- target abstraction value: $dist$

Algorithm:

```

while (true) { // repeat forever
  //find the nearest nodes
  find nodes  $a, b \in G$ :  $\min(distance(a, b))$ 
  // check if nodes are near enough
  if ( $distance(a, b) < dist$ ) { // merge nodes
    create new node  $c \in G$ 
     $c.action = common\_parent(a.action, b.action)$ 
     $c.role = common\_parent(a.role, b.role)$ 
     $c.instances = a.instances \cup b.instances$ 
     $c.pos = avg(position\ of\ agents\ in\ c.instances)$ 
    remove  $a, b$  from  $G$ 
  }
  else {
    break // stop the abstraction process
  }
}

```

Output:

- abstract action graph: G_{dist}

3.2.3 Detection of Strategies

When dealing with the observed multi-agent activity it is difficult to decide whether the observed activity represents an instance of a strategic plan or is it simply a result of agent activity based on some local agent decisions. However, multiple and similar agent activities observed over a longer time period tend to be more strategic than those emerged as local decisions, which can be regarded as noise. Therefore we consider repeated and similar action activity as a strategic activity.

In MASDA, similar actions are represented as a single node in an abstract action graph. The amount of similarity is regulated with the value of abstraction parameter $dist$. Multi-agent activity is represented as a path in an AAG. Hence, repeated activity can be discovered by finding adequate number of complete action instance sequences in an AAG path.

An example of a discovered path, together with traces of its action instances, is shown in Figure 7. Although not all actions continue throughout the whole path, there are three action sequences in this example that span the whole path. As this path exhibits repeated and similar agent activity, it is therefore considered as a representation of a strategic activity.

3.3. Symbolic description generation

This section describes the process of generating symbolic description from the obtained graphic representation of strategic activity.

3.3.1 Action Description Generation

We showed that path nodes from the obtained graphic representation of strategic activity represent a sequence of strategic actions. A symbolic action description is generated by visiting the nodes in a path and extracting role and action concepts together with node position and abstraction value. For example, an obtained strategic action description sequence presented in Table 1 corresponds to AAG_8 path shown in Figure 7. The sequence is strategic because it describes a strategic soccer situation - a successful shot. Again, it is expected that a strategy generated from AAG_{dist} with a greater $dist$ value is more abstract than the one generated from AAG_{dist} with lower $dist$ value.

Action descriptions can be used to create a pattern to detect strategic activity. This is particularly useful when observing unseen agent activity. Namely, if opponent activity matches a starting part of previously constructed strategic action description, it is reasonable to conclude that agents will follow this strategic activity. Consequently, agents can anticipate further opponent actions and devise a suitable counter-activity.

LTeam.FW: Long-dribble (x: 2.3, y: -11.9) dist: 8	LTeam.FW: Pass-to-player (x: 39.2, y: -10.8) dist: 8	LTeam.FW: Successful-shoot (x: 41.0, y: -4.9) dist: 8	LTeam.FW: Successful-shoot (End) (x: 53.6, y: 4.5) dist: 8
--	---	--	--

Table 1: A strategic action description from AAG_8

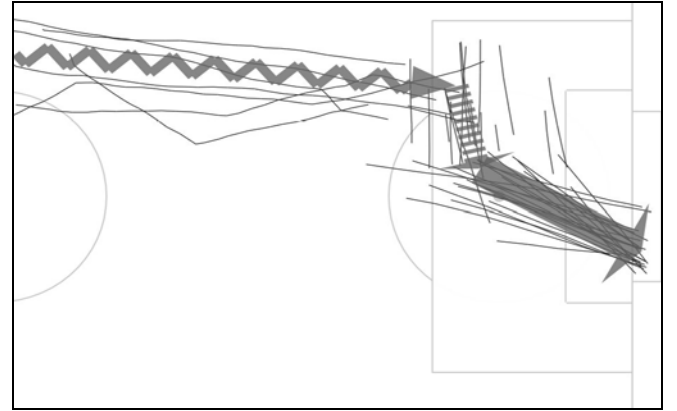


Figure 7: Graphical representation of strategic activity.

3.3.2 Dataset Generation and Rule Induction

As each node defines unique action concept it can be used to generate rules that describe this specific agent action. We generate data suitable for rule induction algorithms as follows. Positive examples are action instances in the target node and negative examples are instances in nearby nodes (i.e. near misses). For each time step we generate all pairs (*agent role*, *domain feature*), written as *role:feature*, and store only the true ones. However, the standard attribute-value approach is not suitable here. Namely, agents dynamically change roles, and thus it is impossible to generate all *role:feature* pairs for all time steps. Consider an example with only positive examples describing an action X. Internal data representation is presented in Table 2 with its corresponding attribute-value representation shown in Table 3.

		agentA		agentB		
time	role	feature1	feature2	role	feature1	feature2
1	L-FW	T	T	C-MF	F	T
2	L-FW	T	F	C-MF	T	T
3	C-MF	F	T	R-FW	T	T
4	C-MF	F	T	R-FW	T	F

Table 2: Internal data representation

		L-FW		R-FW		C-MF	
time		feature1	feature2	feature1	feature2	feature2	feature2
1		T	T	/	/	F	T
2		T	F	/	/	T	T
3		/	/	T	T	F	T
4		/	/	T	F	F	T

Table 3: Attribute-value data representation with roles instead of agent names

Because *role:feature* pairs are not defined for all times, this representation suffers from poor expressiveness. A more appropriate approach is to define *set-valued attributes* that are attributes whose domains are sets instead of single values. With this approach, each feature corresponds to one set-valued attribute while the values are sets of agent roles, whose corresponding *agent:feature* pair is true. The example data is presented in Table 4.

time	feature1	feature2
1	{L-FW}	{L-FW, C-MF}
2	{L-FW, C-MF}	{C-MF}
3	{R-FW}	{R-FW, C-MF}
4	{R-FW}	{C-MF}

Table 4: Set-valued attribute data representation

Due to hierarchically ordered nature of role concepts, it is possible to automatically generate common parent role concepts, as shown in Table 5. This allows rule inducer to induce rules that describe agent activity in a stricter and more abstract manner. Namely, a rule generated from example shown in Table 4 is *IF (C-MF:feature2) $\Rightarrow$ X* while rule generated from example in Table 5 is *IF (FW:feature1 $\wedge$ C-MF:feature2) $\Rightarrow$ X*. The latter rule exhibits more abstract role concept (FW instead of L-FW and R-FW) and is more specific, as it has two features instead of only one as in the previous rule.

time	feature1	feature2
1	{L-FW, FW}	{L-FW, C-MF}
2	{L-FW, C-MF, FW}	{C-MF}
3	{R-FW, FW}	{R-FW, C-MF}
4	{R-FW, FW}	{C-MF}

Table 5: Set-valued attribute data representation with generated common parent role concepts (FW)

4. OBTAINED RESULTS

We generated data for each action obtained from nodes in a discovered strategic path. We used SLIPPER [4], a set-valued rule inducer, with fixed values of parameters to generate corresponding rules. In a typical experiment, 10 games were given as input and several tens of strategic action descriptions

were constructed each time. For a chosen description we constructed rules for AAG_1 to AAG_{20} .

The example in Table 6 presents automatically constructed rules generated from AAG_8 action description (see Figure 8).

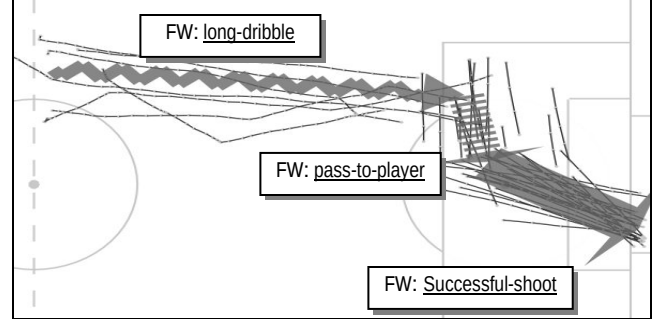


Figure 8: Discovered strategic action description (thick arrows) and original action sequence instances (thin lines).

LTeam.FW:Long-dribble (First frame in Figure 1): RTeam.C-MF:Moving-away-slow $\wedge$ RTeam.L-FB:Still $\wedge$ RTeam.R-FB:Short-distance
LTeam.FW:To-player (Second frame in Figure 1): RTeam.R-FB:Immediate
LTeam.FW:Successful-shoot (Third frame in Figure 1): RTeam.C-FW:Moving-away $\wedge$ LTeam.R-FW:Short-distance
LTeam.FW:Successful-shoot-END (Last frame in Figure 1): RTeam.RC-FB:Left $\wedge$ RTeam.RC-FB:Moving-away-fast $\wedge$ RTeam.R-FB:Long-distance

Table 6: Rules describing discovered strategy

At a first glance the rules seem a bit uninformative. However, expert analysis revealed that the rules do describe important strategic information:

- **LTeam.FW:Long-dribble**

RTeam.C-MF:Moving-away-slow: The right team's center-midfielder was occupied with previous ball owner and was for a while moving away from the ball. Consequently, he stayed behind the action.

RTeam.L-FB:Still: This player is at the bottom of the first frame in Figure 1 moving adjacent to the ball. In terms of relative movement to the ball he appears still. His movement is defensive, but he fails to aim to directly intercept or cover any of the attackers.

RTeam.R-FB:Short-distance: The right team's right fullback player unsuccessfully tried to intercept the dribbler. He consequently stayed at short distance behind the ball. Humans would probably favor the equivalent *RTeam.R-FB:behind* feature.

The three clauses successfully point out three main reasons for the success of the attack.

- **LTeam.FW:To-player**

RTeam.R-FB:Immediate: This rule essentially states that a pass occurred because right team's right fullback came too close to the ball. A ball owner was threatened by the opponent so he passed the ball to the teammate, who was left uncovered.

▪ LTeam.FW:Successful-shoot

RTeam.C-FW:Moving-away: Since shots represent a very fast ball movement, players are moving away from the ball. The rule inducer therefore just picked one feature among many equivalents. This is essentially the same as *ball:fast* which would have been far more intuitive.

LTeam.R-FW:Short-distance: there are many agents at short ball distance (*LTeam.L-FW*, *LTeam.R-FW*, *RTeam.L-FB*, *RTeam.LC-FB*, *RTeam.RC-FB*, and *RTeam.R-FB*). As rule inducer considers shorter rules to be better than longer, it only included one such feature instead of more, as would be expected by humans.

It is evident that rule inducer is not able to choose the most appropriate human-favored features. One reason for this is in a small number of learning examples and consequently substantial number of equivalent *agent:feature* pairs for each situation. As rule inducer has no meanings to favor one feature over another, it simply picks one. However, expert analysis showed that this nevertheless results in meaningful rules describing actual causal strategic relations. In terms of human comprehension, it appears that rule inducer would greatly benefit from some sort of domain-dependent knowledge about features. We consider the construction of such feature rating mechanism as our future work.

5. EXPERIMENTAL EVALUATION

We evaluated MASDA on 10 RoboCup games played during SSIL [14]. A leave-one-out strategy was used to generate 10 learning tasks. A pre-determined 3-action strategy, shown in Table 1 and in Figure 7, was used as a reference and was generated from all 10 games, for abstractions AAG₁ to AAG₂₀. For each learning task, a strategy was generated on 9 games and tested on the remaining game, again for AAG₁ to AAG₂₀. Tests measured the difference between action descriptions, the quality of generated rules for all three actions and joint use of rules together with action descriptions. One learning or test example is one 3-action sequence which is positive if it fits the reference strategy description. If the positive test example fits the learned strategy then this example is considered a true positive, otherwise it is considered a false negative. The results presented in Table 8 are averaged over 10 tests where the x-axis is the level of abstraction (parameter *dist* for AAG_{dist}).

The results in Table 8a indicate that the accuracy of action descriptions is approximately constant with the abstraction. However, the accuracy of rules quickly increases until it peaks at *dist*=4 and then slowly decreases.

This is expected because for lower abstractions, nodes represent only a few action instances, which is insufficient for rule inducer to generate good rules. This is different with high *dist* values where nodes represent quite different action concepts, thus producing more abstract and less accurate rules. Accuracy when using both rules and description follows rule accuracy for lower abstraction values, since rules are too specific. With higher abstraction values (>14) overall accuracy follows description accuracy since rules become too general.

When measuring recall, all the three test scenarios give similar results as shown in Table 8b. That is, the quality of classifying true cases increases to *dist*=3 then remains steady until *dist*=14 when it quickly drops. The similar phenomenon is observed when measuring precision, shown in Table 8c. This can be explained by the generated agent behavior being too abstract.

The last test was performed to verify if the abstraction process really generates more abstract descriptions. For this test we measured the abstraction of generated rules, defined as an average feature depth in feature taxonomy for features used in rules. The lower the depth, the more abstract the rule is. The results, presented in Table 7, clearly show that average feature depth is negatively correlated with the abstraction parameter *dist*. This confirms that as abstraction value increases the rules contain more abstract features. Since rules were generated from node data, we can safely conclude that generated strategies and AAGs exhibit the same principle: that the AAG with higher *dist* value presents more abstract action activity than the one with lower *dist* value.

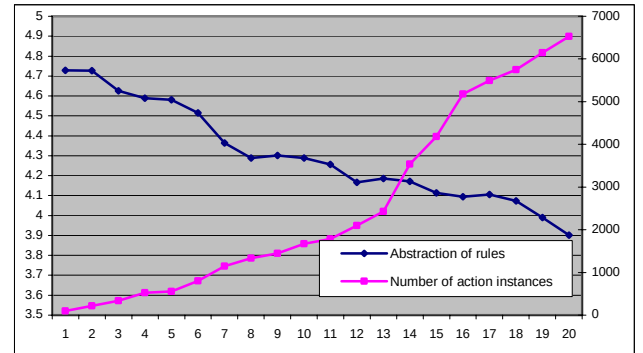


Table 7: The abstraction of generated rules and number of action instances in one strategy

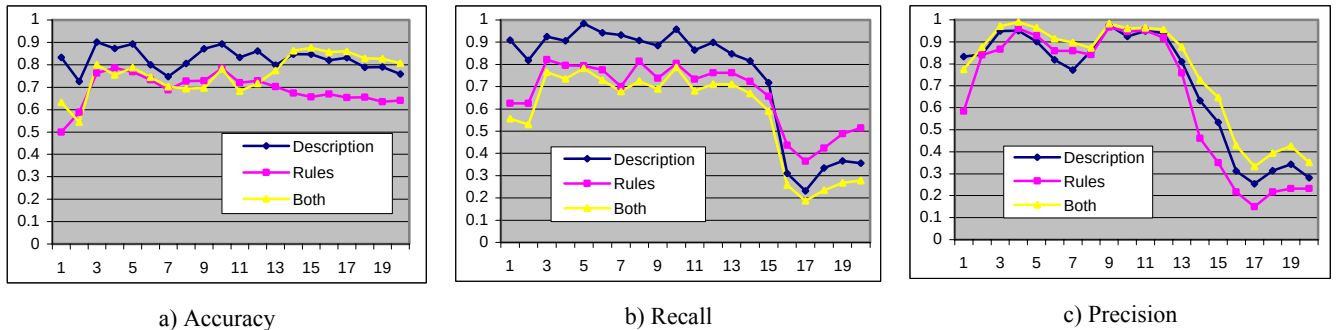


Table 8: Results. The x-axis is the degree of abstraction (parameter *dist*).

6. DISCUSSION

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

The experiments indicate that the approach is practically useful in terms of accuracy, recall and precision. Our tests also confirm that the abstraction process in fact generates more abstract descriptions of agent activity.

There are some open questions that need to be addressed. First, MASDA was evaluated only on the RoboCup domain. Although the authors believe that no major problem should emerge when introducing another domain, this should be verified in practice. Second, while the discovered rules seemed sensible to the soccer expert, they were rather unintuitive for non-experts to comprehend. The rule discovery process would greatly benefit from some sort of feature relevance function, based on domain-knowledge that would favor human-desired features. This kind of knowledge should be provided by a domain expert.

7. ACKNOWLEDGEMENT

The authors would like to thank Gal Kaminka for helpful discussion and thoughtful comments.

REFERENCES

- [1] A. Bezek: *Modeling Multiagent Games Using Action Graphs*. Proceedings of Modeling Other Agents from Observations (MOO 2004), 2004.
- [2] A. Bezek: *Discovering Strategic Multi-Agent Behavior in a Robotic Soccer Domain*, Proceedings of AAMAS 05, 2005.
- [3] J. Calmet, A. Daemi: *From Entropy to Ontology*. Proceedings of Fourth International Symposium From Agent Theory to Agent Implementation, 2004.
- [4] 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.
- [5] S. Hirano in 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), 2004.
- [6] G. Kaminka in D. Avrahami: *Symbolic Behavior-Recognition*. Proceedings of Modeling Other Agents from Observations (MOO 2004), 2004.
- [7] G. Kaminka, M. Fidanboyulu, A. Chang, in M. Veloso: *Learning the sequential coordinated behavior of teams from observations*. Proceedings of the RoboCup-2002 Symposium, 2002.
- [8] D. Koller and B. Milch: *Multi-Agent Influence Diagrams for Representing and Solving Games*. Games and Economic Behavior, 45(1), pp. 181-221, 2003.
- [9] G. Kuhlmann, P. Stone, and J. Lallinger: *The UT Austin Villa 2003 Champion Simulator Coach: A Machine Learning Approach*. In Daniele Nardi, et al., editors, RoboCup-2004: Robot Soccer World Cup VIII, Springer Verlag, Berlin, 2005.
- [10] R. Nair, M. Tambe in 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, 2003.
- [11] S. S. Intille: *Visual Recognition of Multi-Agent Action*. PhD Thesis, MIT, 1999.
- [12] 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.
- [13] T. Raines, M. Tambe, and S. Marsella: *Towards automated team analysis: a machine learning approach*. Third international RoboCup competitions and workshop, 1999.
- [14] RoboCup 2004: *RoboCup Simulation League*. RoboCup04 world cup game repository, <http://carol.science.uva.nl/~jellekok/robocup/rc04/>, 2004.
- [15] T. Steffens et. al: *RoboCup Special Interest Group on Multi-Agent Modelling*. <http://www.cogsci.uni-osnabrueck.de/~tsteffen/mamsig/>, 2001.
- [16] G. Sukthankar and K. Sycara: *A cost minimization approach to human behavior recognition*. Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, The Netherlands, ACM Press, 2005.
- [17] D. Suryadi and P. J. Gmytrasiewicz: *Learning models of other agents using influence diagrams*. Proceedings of the Int. Conference on User Modeling, pp. 223-232, 1999.