Sequential Pattern Mining for Situation and Behavior Prediction in Simulated Robotic Soccer

Andreas D. Lattner, Andrea Miene, Ubbo Visser, and Otthein Herzog

Center for Computing Technologies – TZI, Universitaet Bremen,
PO Box 330 440, D-28334 Bremen, Germany
{adl, andrea, visser, herzog}@tzi.de

Abstract. Agents in dynamic environments have to deal with world representations that change over time. In order to allow agents to act autonomously and to make their decisions on a solid basis an interpretation of the current scene is necessary. If intentions of other agents or events that are likely to happen in the future can be recognized the agent's performance can be improved as it can adapt the behavior to the situation. In this work we present an approach which applies unsupervised symbolic learning off-line to a qualitative abstraction in order to create frequent patterns in dynamic scenes. These patterns can be later applied during runtime in order to predict future situations and behaviors. The pattern mining approach was applied to two games of the 2D RoboCup simulation league.

1 Introduction

Agents in dynamic environments have to deal with world representations that change over time. In order to allow agents to act autonomously and to make their decisions on a solid basis an interpretation of the current scene is necessary. Scene interpretation can be done by checking if certain patterns match the current belief of the world. If intentions of other agents or events that are likely to happen in the future can be recognized the agent's performance can be improved as it can adapt the behavior to the situation. One step into this direction is the recognition of formations [24].

We focus on qualitative representations as they allow a concise representation of the relevant information. Such a representation provides means to use background knowledge, to plan future actions, to recognize plans of other agents, and is comprehensible for humans the same time. In our approach we map quantitative data to qualitative representations. We use time series which are divided into different segments satisfying certain monotonicity or threshold conditions [18, 19]. One example is that if the distance between two objects is observed it can be divided into increasing and decreasing distance representing approaching and departing relations (cf. [19]).

Additionally to the requirement to handle situations which change over time, relations between arbitrary objects can exist in their belief of the world. In this work we present an approach which applies unsupervised symbolic learning to a

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 118–129, 2006.

qualitative abstraction in order to create frequent patterns in dynamic scenes. Here, we apply an extended version of the sequential pattern mining algorithm presented in [13] to data from simulated robotic soccer games of the 2D RoboCup simulation league. We propose the application of the learned rules to predict future situations and behavior in order to support behavior decision.

2 Related Work

Association rule mining addresses the problem of discovering association rules in data. One famous example is the mining of rules in basket data [1]. Different algorithms have been developed for the mining of association rules in item sets (e.g., [2]). Mannila et al. extended association rule mining by taking event sequences into account [15]. They describe algorithms which find all relevant episodes which occur frequently in the event sequence. H"oppner presents an approach for learning rules about temporal relationships between labeled time intervals [8]. The labeled time intervals consist of propositions. Relationships are described by Allen's interval logic [3]. Other researchers in the area of spatial association rule mining allow for more complex representations with variables but do not take temporal interval relations into account (e.g., [11, 14, 16]).

The learning approach presented here combines ideas from different directions. Similar to H"oppner's work [8] the learned patterns describe temporal interrelationships with interval logic. Contrary to H"oppner's approach our representation allows for describing predicates between different objects similar to approaches like [14]. The generation of frequent patterns comprises a top-down approach starting from the most general pattern and specializing it. At each level of the pattern mining just the frequent patterns of the previous step are taken into account knowing that only combinations of frequent patterns can result in frequent patterns again which is a typical approach in association rule mining (e.g., [15]).

RoboCup is used as an application domain for learning approaches in different papers (e.g., [10, 12, 17, 21, 22, 23, 25]). Many of the recent approaches apply reinforcement learning to robotic soccer.

Riley and Veloso [20] use a set of pre-defined movement models and compare these with the actual movement of the players in set play situation. In new set play situations the coach then uses the gathered information to predict the opponent agent's behavior and to generate a plan for his own players. The approach uses probabilistic models and can be used both off-line and on-line.

Frank and colleagues [5] presented a real-time approach which is based on statistical methods. The approach gathers information such as the percentage of ball-holding of a certain player or which player passes the ball to which team mate. The result is a thorough statistical analysis which can then be used to derive information about a game being played. This can help for new future developments of a team.

A hybrid approach to learn the coordinated sequential behavior of teams was presented by Kaminka and colleagues [10]. The idea is to take time-series

of continuous multi-variate observations and then parse and transform them into a single-variable categorial time-series. The authors use a set of behavior recognizers that focus only on recognizing simple and basic behaviors or the agents (e.g., pass, dribble). The data are then represented in a trie (a tree-like data structure) to support two statistical methods: (a) frequency counting and (b) statistical dependency detection. Experiments showed that the latter method is more suitable to discover sequential behavior.

Huang and colleagues [9] recently published an approach for plan recognition and retrieval for multi-agent systems. The approach is based on observations of agents' coordinative behaviors. The basis are players' element behaviors sequences (e.g., pass, dribble, shoot) which are sorted in a temporal order. The field is decomposed into cells where each cell denotes one agent's behavior at a time slice. Interesting and frequent behavior sequences are considered as the team's plans on the assumption that the team's plan is embedded in those sequences. The discovery of significance of sequence patterns are based on statistical evidences. The promising results are plans based on observation.

Most similar to our work are the approaches of Kaminka et al. [10] and Huang et al. [9] as they also create a sequence of certain events or behaviors and search for frequent sequences. The main difference to our approach is the representational power of the learned patterns. Our representation allows for using variables in the learned rules and for identifiying arbitrary temporal relations between predicates (e.g., like Allen's interval logic).

3 Qualitative Motion Description

Our approach to qualitative motion description maps quantitative data to qualitative representations. Time series are divided into intervals which satisfy certain monotonicity or threshold conditions [19, 18]. Time series include absolute motion of single objects and relative motion of pairs of objects, which is described by changes in their pairwise spatial relations over time.

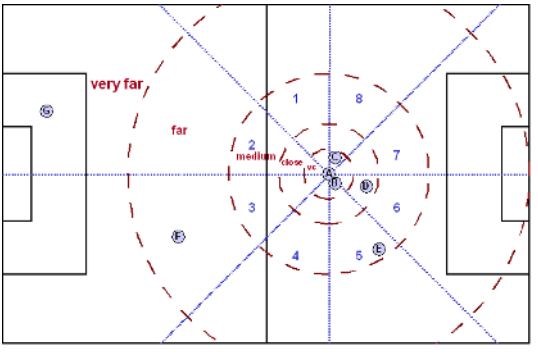

Fig. 1. Distance and direction classes

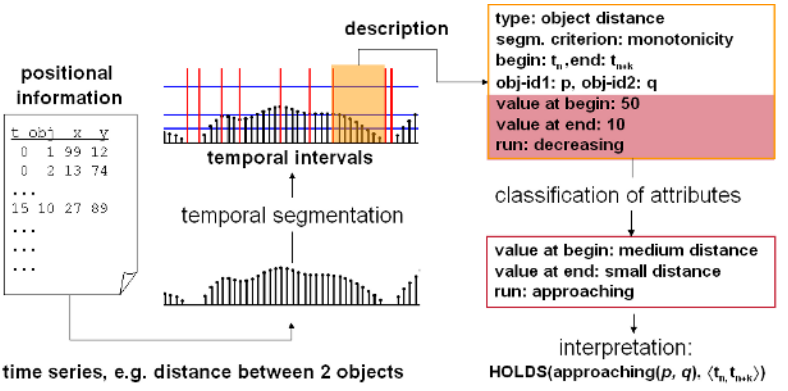

Fig. 2. Generation of motion description

On a quantitative level the objects' absolute and relative motion is described by four types of time series: the motion direction and speed of each object, and the spatial direction and distance for each pair of objects. In a first abstraction step each time series is segmented into time intervals of homogeneous motion values. We use two different segmentation methods: a threshold-based segmentation method, which represents the values of an interval by their average value and a monotonicity-based segmentation method, which groups together increasing, decreasing and constant values sequences. This preserves the dynamic of motion and allows for the description of dynamic aspects as, e.g., acceleration or approaching/departing. In a second step the attribute values describing the intervals are mapped onto qualitative classes for direction, speed or distance, respectively (see Fig. 1). We distinguish eight direction classes and five speed respective distance classes, which are organized in distance systems [7]. The radius of each distance class is double the size of the radius of the previous one. For the soccer domain we use the heading of each player as reference axis for the representation of the spatial relations to the surrounding objects which leads to an egocentric point of view.

Fig. 2 shows an example of the entire process of motion description for a time series of object distances, segmented by the monotonicity-based segmentation criterion.

The interval shown in the example in Fig. 2 is interpreted as an approaching of the objects p and q, which is expressed by the term HOLDS(approaching(p, q), $\langle t_n, t_{n+k} \rangle$). The predicate HOLDS expresses the coherence between a certain situation (here approaching) and the time interval $\langle t_n, t_{n+k} \rangle$ in which it is taking place or is valid [4].

As input for the learning algorithm described in the following section we use a subset of the motion description including intervals concerning the relations *meets, approaches* and *departs* between a player and the ball. For the relation *approaches* we restrict to intervals in which the two objects approach at least until they are at a medium distance. For the relation *departs* we restrict to intervals in which the two objects are in medium distance or closer when starting

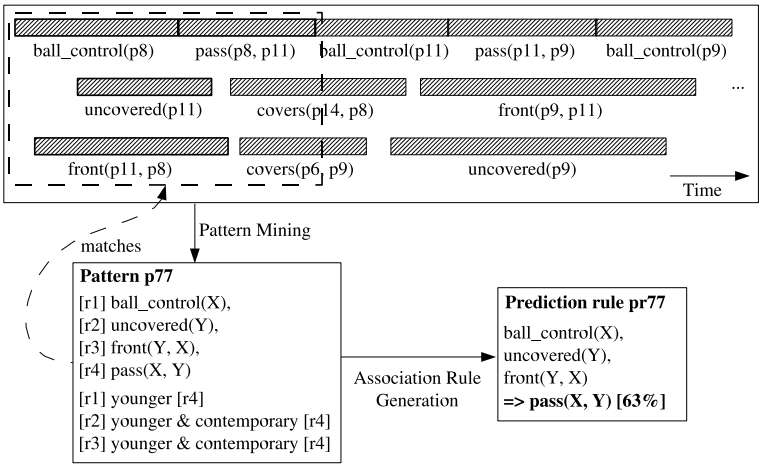

Fig. 3. Pattern and prediction rule generation

departing. Additionally further higher-level relations like *pass* and *ball_control* are extracted from the scenes.

4 Sequential Pattern Mining

Here, a dynamic scene is represented symbolically by a set of objects and predicates between these objects as created by the qualitative abstraction described in the previous section. The predicates are only valid for certain time intervals and the scene can thus be considered as a sequence of (spatial or conceptual) predicates. These predicates are in specific temporal relations regarding the time dimension. An example for such a sequence can be seen at the top of Fig. 3.

Each predicate r is an instance of a predicate definition rd. We use the letter r for predicates/relations; the letter p is used for patterns. $\mathcal{R}_{schema} = \{rd_1, rd_2, \ldots\}$ is the set of all predicate definitions $rd_i := \langle l_i, a_i \rangle$ with label l_i and arity a_i, i.e., each rd_i defines a predicate between a_i objects. Predicates can be hierarchically structured. If a predicate definition rd_1 specializes another predicate definition rd_2 all instances of rd_1 are also instances of the super predicate rd_2.

If we have to handle more than one dynamic scene, let $\mathcal{S} = \{s_1, s_2, \ldots\}$ be the set of the different sequences s_i. A single sequence s_i is defined as $s_i = (\mathcal{R}_i, \mathcal{TR}_i, \mathcal{C}_i)$ where $\mathcal{R}_i$ is the set of predicates, $\mathcal{TR}_i$ is the set of temporal relations and $\mathcal{C}_i$ is the set of constants representing different objects in the scene. Every constant is an instance of a class (default is the top concept "object") and classes form an inheritance hierarchy. Each predicate is defined as $r(c_1, \ldots, c_n)$ with r being an instance of $rd_i \in \mathcal{R}_{schema}$, having arity $n = a_i$, and $c_{i,1}, \ldots, c_{i,n} \in \mathcal{C}_i$ are representing the objects where the predicate holds. The set of temporal relations $\mathcal{TR}_i = \{tr_1, tr_2, \ldots\}$ defines relations between pairs of elements in $\mathcal{R}_i$. Each temporal relation is defined as

$tr_i(r_a, op, r_b)$ with $r_a, r_b \in \mathcal{R}_i$. op is the set of valid temporal relations. If Allen's temporal relations between intervals [3] are used, this set is defined as $op \in \{<, =, >, d, di, o, oi, m, mi, s, si, f, fi\}$. It is also possible to use other temporal relations, e.g., those defined by Freksa [6].

4.1 Pattern Representation and Pattern Matching

Patterns are abstract descriptions of sequence parts with specific properties. A pattern defines what predicates must occur and how their temporal interrelationship has to be. Let $\mathcal{P} = \{p_1, p_2, \ldots\}$ be the set of all patterns p_i. A pattern is (similar to sequences) defined as $p_i = (\mathcal{R}_i, \mathcal{TR}_i, \mathcal{V}_i)$.

$\mathcal{R}_i$ is the set of predicates $r_{ij}(v_{ij,1}, \ldots, v_{ij,n})$ with $v_{ij,1}, \ldots, v_{ij,n} \in \mathcal{V}_i$. $\mathcal{V}_i$ is the set of all variables used in the pattern. A class is assigned to each variable. $\mathcal{TR}_i$ defines the set of the temporal relations which have already been defined above.

A pattern p matches in a (part of a) sequence sp if there exists a mapping of a subset of the constants in sp to all variables in p such that all predicates defined in the pattern exist between the mapped objects and all time constraints of p are satisfied by the time intervals in the sequence without violating the class restrictions. In order to restrict the exploration region a window size can be defined. Only matches within a certain neighborhood (specified by the window size) are valid.

During the pattern matching algorithm a sliding window is used, and at each position of the window all matches for the different patterns are collected. A match consists of the position in the sequence and an assignment of objects to the variables of the pattern. Fig. 3 illustrates a sample pattern and one of the matches in the given sequence. In this example temporal relations as defined by Freksa [6] are used.

4.2 Pattern Generation

Different patterns can be put into generalization-specialization relations. A pattern p_1 subsumes another pattern p_2 if it covers all sequence parts which are covered by p_2: $p_1 \sqsubseteq p_2 := \forall sp, matches(p_2, sp) : matches(p_1, sp)$.

If p_1 additionally covers at least one sequence part which is not covered by p_2 it is more general: $p_1 \sqsubset p_2 := p_1 \sqsubseteq p_2 \wedge \exists sp_x : matches(p_1, sp_x), \neg matches(p2, sp_x)$. This is the case if $p_1 \sqsubseteq p_2 \wedge p_1 \not\supseteq p_2$.

In order to specialize a pattern it is possible to add a new predicate r to $\mathcal{R}_i$, add a new temporal relation tr to $\mathcal{TR}_i$, specialize the class of a variable, unify two variables, or specialize a predicate, i.e., replacing it with another more special predicate. Accordingly it is possible to generalize a pattern by removing a predicate r from $\mathcal{R}_i$, removing a temporal relation tr from $\mathcal{TR}_i$, inserting a new variable, or generalizing a predicate r, i.e., replacing it with another more general predicate.

At the specialization of a pattern by adding a predicate a new instance of any of the predicate definitions can be added to the pattern with variables which are not used in the pattern so far. Specializing the class of a variable means that the

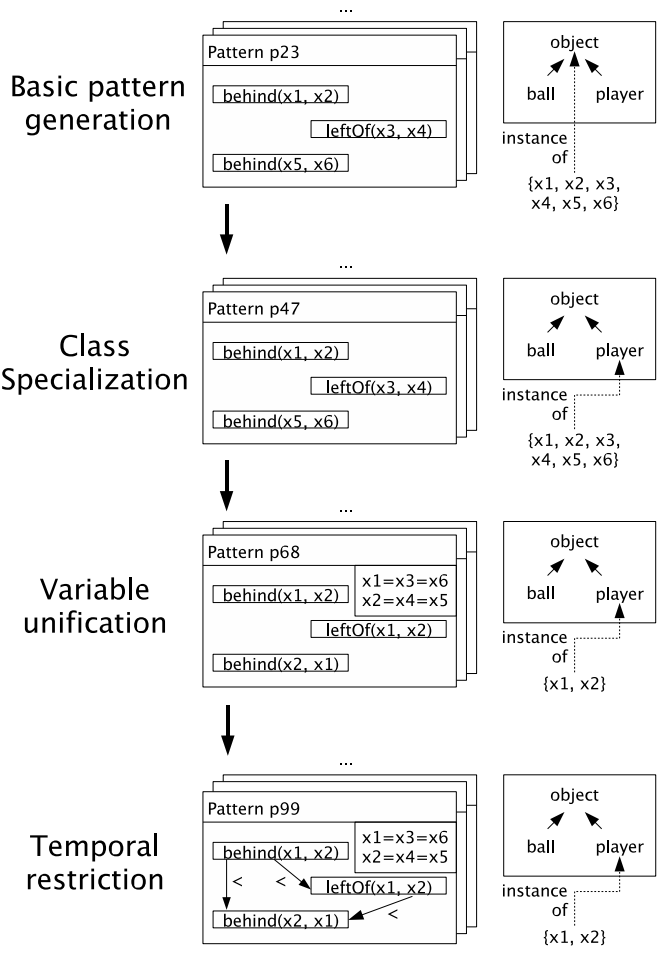

Fig. 4. Pattern generation

current class assigned to a variable is replaced by one of its subclasses. If a pattern is specialized by adding a new temporal relation for any pair of predicates in the pattern (which has not been constrained so far) a new temporal restriction can be added. A specialization through variable unification can be done by unifying an arbitrary pair of variables, i.e., the different predicates can be "connected" via identical variables after this step. Another specialization would be to replace a predicate by a more special predicate. These different specialization steps can be used at the top-down generation of rules (see Fig. 4).

In order to evaluate the patterns (and to just keep the k best patterns in the steps of the pattern learning algorithm) an evaluation function was defined. Right now our evaluation function takes six different values into account: relative pattern size, relative pattern frequency, coherence, temporal restrictiveness, class restrictiveness, and predicate preference. Further criteria can be added if

necessary. The overall evaluation function of a pattern computes a weighted sum of the different single criteria.

The coherence gives information how "connected" the predicates in the pattern are by putting the number of used variables in relation to the maximum possible number of variables for a pattern. The temporal restrictiveness is the number of restricted predicate pairs in the pattern to the maximum number of time restrictions. The class restrictiveness is the relation of the number of variables which cannot be specialized anymore to the total number of variables. Further information about the evaluation criteria can be found in [13].

Compared to the work presented in [13] the representational power was extended by introducing classes and allowing temporal parallelism of time points and limiting the patterns to keep by discarding those patterns which create more special patterns without loss of frequency. With these extensions it is possible to restrict variables in patterns to specific classes, e.g., indicating that an object must be a player or a ball. Allowing identical time points for start and end time points of different predicates extends the number of learnable temporal relations (e.g., *meets* and *starts*; cf. [3]). Discarding general patterns which are not more frequent than their specialized patterns reduces complexity in further steps of the algorithm because the overall number of kept pattern decreases.

4.3 Situation and Behavior Prediction

Fig. 5 illustrates the idea how to use sequential pattern mining for situation and behavior prediction. We assume that the quantitative data perceived from the sensors is mapped to a qualitative representation. This should be a concise representation with all relevant information for behavior decision where similar

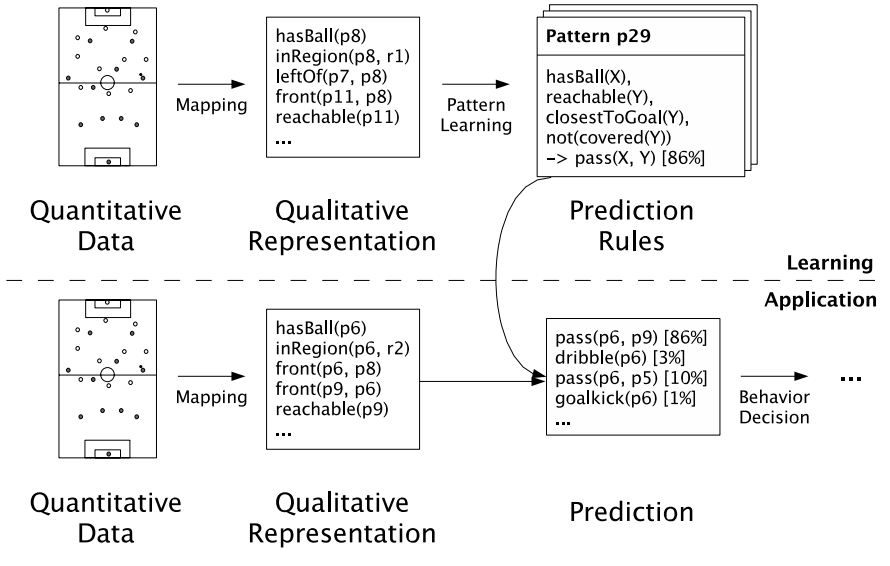

Fig. 5. Pattern learning for situation and behavior prediction

(quantitative) information is mapped to one qualitative class. This can include various information, e.g., about distances, regions, ball possession, score, time etc. This information is used as input for the learning algorithm. The start and end time points where different predicates are valid are known and can be used to set up a (temporal) sequence of predicates describing a dynamic scene. In the current setting learning is performed off-line. The result of learning is a set of prediction rules which give information what (future) actions or situations might occur with some probability if certain preconditions are satisfied. During a game these rules can be applied in order to predict what future actions or relations are likely to happen. This information can be used to perform a behavior decision on a better basis.

5 Application to Simulated Robotic Soccer

In the current status the qualitative mapping and the pattern learning approach are realized. The automated extraction and application of behavior prediction rules has to be done in future work. In order to evaluate our approach soccer games of the RoboCup 2D simulation league (TsinghuaAeolus vs. FC Portugal at June 22nd 2002 and FC Portugal vs. Puppets at June 21st 2002) were analyzed and a qualitative motion description as presented in Section 3 was created.

```
s; closerToGoal; 9;  8; 1
s; closerToGoal; 9; 11; 1
s; front;        9;  7; 1
s; front;        9; 10; 1
s; front;        9;  2; 2
s; front;        9;  3; 2
e; front;        9;  2; 2
e; front;        9;  3; 2
s; closerToGoal; 9; 10; 3
s; closerToGoal; 9;  7; 4
e; front;        9;  7; 6
e; front;        9; 10; 6
s; front;        9;  8; 8
...
```

```
s; pass;         16; 18; 338
e; front;        16; 18; 338
s; closerToGoal; 18; 19; 341
s; closerToGoal; 18; 20; 341
s; closerToGoal; 18; 21; 341
s; closerToGoal; 18; 22; 341
s; front;        18; 12; 341
s; front;        18; 15; 341
e; closerToGoal; 16; 18; 341
e; closerToGoal; 16; 19; 341
e; closerToGoal; 16; 20; 341
e; closerToGoal; 16; 21; 341
e; closerToGoal; 16; 22; 341
...
```

Fig. 6. Sample input for the pattern mining algorithm

```
P_137
 front(X_7 X_8 )[r_1]
 front(X_15 X_16 )[r_2]
 pass(X_19 X_20 )[r_3]
 closerToGoal(X_25 X_26 )[r_4]

 [ X_7 = X_19; X_8 = X_20; X_15; X_16; X_25; X_26;]
 [X_7 = player X_8 = player X_15 = player X_16 = player
 X_19 = player X_20 = player X_25 = player X_26 = player ]
 [r_1 older r_2, r_1 younger-equal r_3
  r_1 younger-equal r_4, r_2 older r_3 ]
```

Fig. 7. Learned pattern

```
[r1] hasBall(X_1),
[r2] front(X_1, X_2),
[r3] uncovered(X_2)
 => [r4] pass(X_1, X_2)
[r1] older [r4],
[r2] older [r4],
[r3] older [r4]
```

Fig. 8. Example for a prediction rule

Fig. 6 shows a sample input for the learning algorithm. This input represents a sequence of predicates with their start and end time points. The first column identifies if a relation starts ("s") or ends ("e"). The second column is the name of the predicate (e.g., `pass`). The identifiers of the objects for which the predicate holds are the third and fourth value in each line. The last column denotes the time stamp of the start or end time point of this relation. Temporal parallelism can be recognized by identical values in the last column.

The learning algorithm performs a top-down generation of patterns. During learning the predicates in the input file are used to generate new patterns with variables. One example of a learned pattern can be seen in Fig. 7. It consists of four predicates (`front` [2x], `pass`, and `closerToGoal`). Among other information it says that a player (`X_7`/`X_19`) is in front of another (`X_8`/`X_20`) after a pass between those two was performed. The three segments below the predicates show the restrictions w.r.t. variable unification (`X_7 = X_19`), class information (`X_7 = player`), and temporal relations (`r_1 older r_2`). This pattern was learned from a snippet of the sequence from the first game. Fig. 8 shows an example of a prediction rule.

6 Conclusion

In this paper we presented an approach to situation and behavior prediction. A sequential pattern mining algorithm is applied in order to learn frequent patterns in the data. These patterns are then transformed into prediction rules which can be applied to estimate what is likely to happen in the future. One characteristic of the learning approach is high representational power with the potential of learning complex patterns with predicates and variables from relational and temporal data.

The drawback of the approach is the high complexity of the learning algorithm as discussed in [13]. The experiments support the assumption that without limiting the search space during pattern generation the algorithm cannot be used to learn complex patterns on-line due to time and space complexity. It is necessary to develop heuristics which allow an efficient learning of patterns without cutting off a large number of potentially good patterns.

We are currently working on improvements in order to handle the complexity of the learning task. In future work the performance of the learned patterns for predicting future behaviors and situations must be analyzed.

Acknowledgment

The work in this paper was partially funded by the Deutsche Forschungsgemeinschaft (DFG) in the SPP-1125.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499, September 1994.
3. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.
4. J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.
5. I. Frank, K. Tanaka-Ishi, K. Arai, and H. Matsubara. The statistics proxy server. In T. Balch, P. Stone, and G. Kraetschmar, editors, *4th International Workshop on RoboCup*, pages 199–204, Melbourne, Australia, 2000. Carnegie Mellum University Press.
6. C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227, 1992.
7. D. Hernández, E. Clementini, and P. Di Felice. Qualitative distances. In *Proceedings of COSIT'95, LNCS 988*, Semmering, Austria, 1995. Springer.
8. F. Höppner. Learning temporal rules from state sequences. In *Proceedings of the IJCAI'01 Workshop on Learning from Temporal and Spatial Data*, pages 25–31, Seattle, USA, 2001.
9. Z. Huang, Y. Yang, and X. Chen. An approach to plan recognition and retrieval for multi-agent systems. In M. Prokopenko, editor, *Workshop on Adaptability in Multi-Agent Systems, First RoboCup Australian Open 2003 (AORC-2003)*, Sydney, Australia, 2003. CSIRO.
10. G. Kaminka, M. Fidanboylu, A. Chang, and M. Veloso. Learning the sequential coordinated behavior of teams from observation. In G. Kaminka, P. Lima, and R. Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI, LNAI 2752*, pages 111–125, Fukuoka, Japan, 2003.
11. K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases, SSD*, pages 47–66, Portland, Maine, 1995.
12. G. Kuhlmann and P. Stone. Progress in 3 vs. 2 keepaway. In *RoboCup 2003: Robot Soccer World Cup VII*. Springer, Berlin, 2004.
13. A. D. Lattner and O. Herzog. Unsupervised learning of sequential patterns. In *ICDM 2004 Workshop on Temporal Data Mining: Algorithms, Theory and Applications (TDM'04)*, Brighton, UK, November 1st 2004.
14. D. Malerba and F. A. Lisi. An ILP method for spatial association rule mining. In *Working notes of the First Workshop on Multi-Relational Data Mining*, pages 18–29, Freiburg, Germany, 2001.
15. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

16. J. Mennis and J. W. Liu. Mining association rules in spatio-temporal data. In *Proceedings of the 7th International Conference on GeoComputation*, University of Southampton, UK, 8 - 10 September 2003.
17. A. Merke and M. Riedmiller. Karlsruhe Brainstormers - a reinforcement learning way to robotic soccer. In *RoboCup 2001: Robot Soccer World Cup V*. Springer, Berlin, 2002.
18. A. Miene, A. D. Lattner, U. Visser, and O. Herzog. Dynamic-preserving qualitative motion description for intelligent vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV '04)*, pages 642–646, June 14-17 2004.
19. A. Miene, U. Visser, and O. Herzog. Recognition and prediction of motion situations based on a qualitative motion description. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII, LNCS 3020*, pages 77–88. Springer, 2004.
20. P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*, number 2377 in Lecture Notes in Artificial Intelligence, pages 453–458, Berlin, 2002. Springer Verlag.
21. P. Riley and M. Veloso. Coaching advice and adaption. In *RoboCup 2003: Robot Soccer World Cup VII, LNCS 3020*, pages 192–204. Springer, Berlin, 2004.
22. P. Stone and R. S. Sutton. Scaling reinforcement learning toward robocup soccer. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
23. P. Stone and M. Veloso. A layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.
24. U. Visser, C. Drücker, S. Hübner, E. Schmidt, and H.-G. Weland. Recognizing formations in opponent teams. In P. Stone, T. Balch, and G. Kraetschmar, editors, *RoboCup 2000: Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Computer Science*, pages 391 – 396, Melbourne, Australia, 2001. Springer-Verlag.
25. U. Visser and H.-G. Weland. Using online learning to analyze the opponent's behavior. In *RoboCup 2002: Robot Soccer World Cup VI, LNAI 2752*, pages 78–93. Springer-Verlag Berlin Heidelberg, 2003.