Mining Temporal Patterns from Relational Data

Andreas D. Lattner and Otthein Herzog
TZI – Center for Computing Technologies, Universität Bremen
PO Box 330 440, D-28334 Bremen, Germany
{adl|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 temporal patterns in dynamic scenes. Here, an adaption of a sequential pattern mining algorithm which was presented earlier by the authors is proposed in order to reduce the complexity by handling different aspects (class restrictions, variable unifications, and temporal relations) separatedly first, and then combining the results of the single steps. The work is still in progress– this paper introduces the basic ideas and shows an example run of the implemented system.

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.

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. Quantitative data has to be mapped to a qualitative representation, e.g., by dividing time series into different segments satisfying certain monotonicity or threshold conditions as suggested by Miene and colleagues [Miene *et al.*, 2004a; 2004b]. 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. [Miene *et al.*, 2004b]).

Additionally to the requirement to handle situations which change over time, relations between arbitrary objects can exist in the belief of the world. In this work we present an approach which applies unsupervised symbolic learning to a qualitative abstraction in order to create frequent patterns in dynamic scenes. In this work an adaption of the sequential pattern mining algorithm presented in [Lattner and Herzog, 2004] is proposed in order to reduce the complexity by handling different aspects (class restrictions, variable unifications, and temporal relations) separatedly first, and then combining the results of the single steps. This work is still in progress, i.e., a detailed evaluation of the approach has to be done in future work. This paper introduces the basic ideas and shows an example run of the system.

2 Related Work

Association rule mining addresses the problem of discovering association rules in data. One typical example is the mining of rules in basket data [Agrawal *et al.*, 1993]. Different algorithms have been developed for the mining of association rules in item sets (e.g., [Agrawal and Srikant, 1994]). Mannila et al. extended association rule mining by taking event sequences into account [Mannila *et al.*, 1997]. They describe algorithms which find all relevant episodes which occur frequently in the event sequence. Höppner presents an approach for learning rules about temporal relationships between labeled time intervals [Höppner, 2001]. The labeled time intervals consist of propositions. Relationships are described by Allen's interval logic [Allen, 1983]. 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., [Koperski and Han, 1995; Malerba and Lisi, 2001; Mennis and Liu, 2003]).

Dehaspe and De Raedt combine association rule mining algorithms with ILP techniques. Their system WARMR is an extension of Apriori for mining association rules over multiple relations [Dehaspe and Raedt, 1997; Dehaspe and Toivonen, 2001]. The generated rules consist of sets of logical atoms. This more expressive representations (compared to itemset mining) allows for discovering rules like: $likes(KID, A), has(KID, B) \Rightarrow prefers(KID, A, B)$ (cf. [Dehaspe and Raedt, 1997]).

The approaches of Kaminka et al. and Huang et al. also create a sequence of certain events or behaviors and search for frequent sequences [Kaminka *et al.*, 2003; Huang *et al.*, 2003]. The main difference to our approach is the representational power of the learned patterns. Our representation allows for using variables (and assigning classes to them) in the learned rules and allows for identifiying arbitrary temporal relations between predicates (e.g., those introduced by [Allen, 1983]).

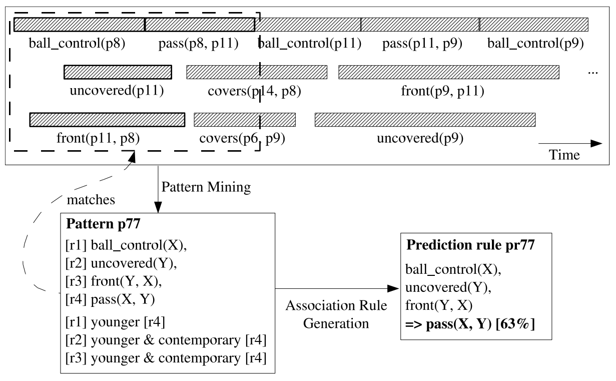

Figure 1: Pattern and prediction rule generation

The learning approach presented here combines ideas from different directions. Similar to Höppner's work [Höppner, 2001] the learned patterns describe temporal interrelationships with interval logic. Contrary to Höppner's approach our representation allows for describing predicates between different objects similar to approaches like [Malerba and Lisi, 2001; Dehaspe and Raedt, 1997; Dehaspe and Toivonen, 2001]. 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., [Mannila *et al.*, 1997]).

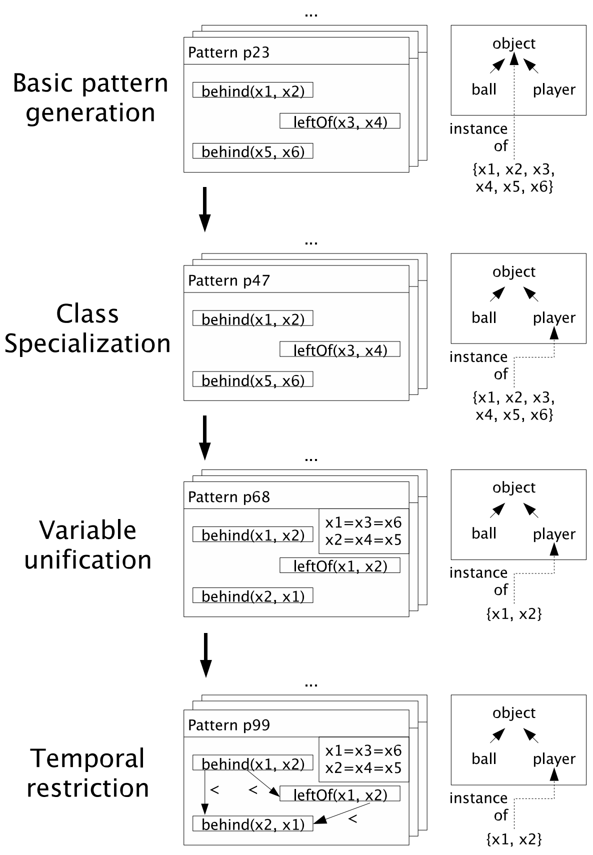

Figure 2: Pattern generation

3 Sequential Pattern Mining

Here, a dynamic scene is represented symbolically by a set of objects and predicates between these objects as e.g. created by the qualitative abstraction described in [Miene *et al.*, 2004a; 2004b]. 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. 1.

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. For each predicate definition it is defined what their ranges are, i.e., it is defined what classes the corresponding objects in predicate instances have to be instances of.

A 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 [Allen, 1983] 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 [Freksa, 1992].

Dehaspe and Toivonen proposed to use a "key parameter" for the support computation [Dehaspe and Toivonen, 2001]. This has the disadvantage that this key predicate must be part of each pattern and not all potentially frequent patterns can be compared. Our notion of support is to count all matches of the pattern in a sequence. As different combinations of (partially identical) predicates can lead to multiple counts we do not allow any predicate to be counted more than once while pattern matching. The current version of the algorithm greedily counts the first match and disables the used predicates of the match for the further pattern matching process.

In the current implementation the matches of the intermediate steps are not stored, i.e., that the pattern matching is done from scratch for each new pattern. A problem here is that with the used support measure it is possible that a match of a previous (more general) pattern might not be extendable by another predicate but a different combination of a subset of the match with other predicates might lead to a match. Due to lack of space this issue cannot be discussed here in detail but will be addressed in future publications.

If more than one sequence has to be processed the support is computed in each sequence separatedly by counting the different pattern matches in the single sequences and accumulating the support values.

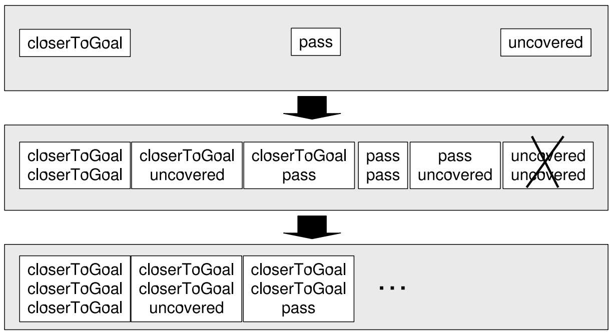
closerToGoal
pass
uncovered
closerToGoal closerToGoal
closerToGoal uncovered
closerToGoal pass
pass pass
pass uncovered
~~uncovered uncovered~~
closerToGoal closerToGoal closerToGoal
closerToGoal closerToGoal uncovered
closerToGoal closerToGoal pass
...

Figure 3: Generation of basic patterns

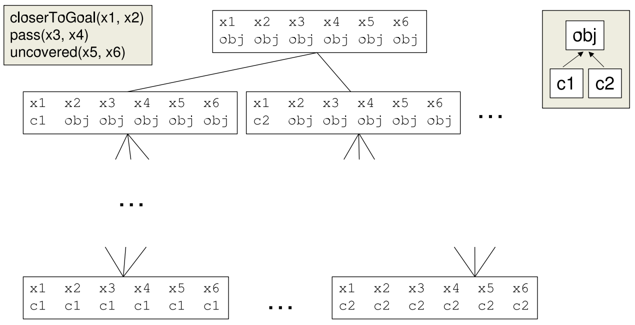

Figure 4: Class lattice

3.1 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 matched predicates in the sequence and an assignment of objects to the variables of the pattern. Fig. 1 illustrates a sample pattern and one of the matches in the given sequence marked by a dashed line. In this example temporal relations as defined by Freksa [Freksa, 1992] are used. The example also illustrates how an association rule could be created from the pattern.

3.2 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 $p_1 \sqsubseteq p_2 \wedge p_1 \not\sqsupseteq p_2$.

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.

Fig. 2 shows the different steps of the pattern generation process. At the specialization of a basic pattern by adding a predicate a new instance of any of the predicate definitions can be added to the pattern with variables which have not been used in the pattern so far. This Apriori-like step of basic pattern generation is illustrated in Fig. 3.

For each basic pattern it is possible to perform further specialications. We take into account specializations by adding different kinds of restrictions w.r.t. classes, varibale unifications, and temporal relations to the basic patterns. These different kinds of specializations can be seen as a search through lattices as illustrated in Fig. 4 - 6. Here, the top-level elements are the most general restrictions while the elements at the bottom (leaves) are the most special restrictions in the lattices.

Specializing the class of a variable means that the current class assigned to a variable is replaced by one of its subclasses (Fig. 4). The background knowledge defines the class hierarchy and at a specialization step the class for a variable is replaced by one of its subclasses.

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 (Fig. 5). In the general case each variable can be bound to an arbitrary constant, i.e., it does not matter if variables are bound to identical or different constants. If a variable restriction is added it is stated that two variables must be bound to the same object, e.g., $x_1 = x_2$ in the first left branch of Fig. 5.

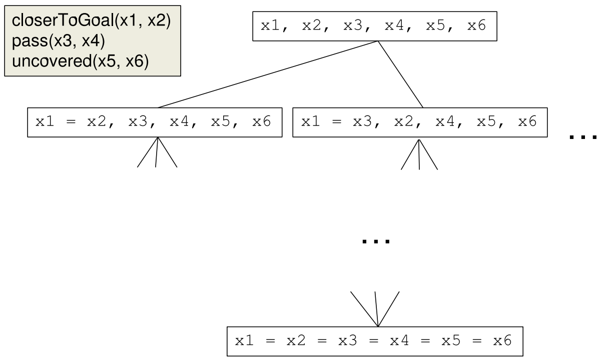

Figure 5: Variable lattice

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. Initially, no temporal restrictions exist, i.e., the time of appearance of certain predicates does not make any

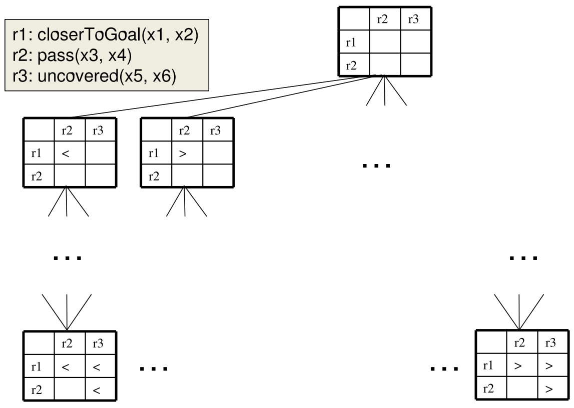

Figure 6: Temporal lattice

difference. Temporal restrictions state that there must exist a certain temporal relation between two predicates, e.g., that one must be *before* the other. In the first left branch of Fig. 6 it is stated that `closerToGoal(x1, x2)` must be *before* (denoted by $<$) `pass(x3, x4)`.

3.3 Generation and Merging of Separate Borders

If all three kinds of restrictions (class, variable, and temporal restrictions) were handled together at the same time this would lead to a huge search space for specializing basic patterns by these restrictions. At each step in the specialization process a great number of restrictions to add could possibly exist. Handling the different kinds of restrictions separated seems to be a good approach in order to reduce complexity. But if for all frequent patterns all frequent restrictions are created and then combined in a brute-force way (i.e., creating all combinations) complexity would make mining impossible even for a small number of predicates.

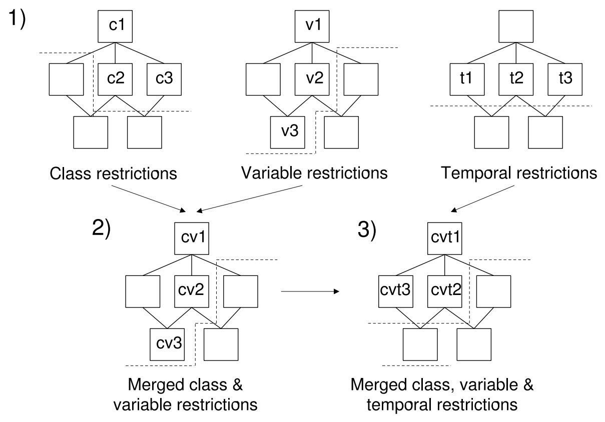

Figure 7: Merging of lattices

Our approach to handling the complexity is to just work with the most special borders in each of the single "dimensions" (i.e., restriction kinds) and then merge this borders in following steps. The algorithms starts from the most special restrictions in each single lattice and generalizes the restrictions until frequent patterns are found[1]. The initial most special borders (independent of a sequence and a pattern's support in it) can be generated by collecting the leaves in the created lattices. The actual most special borders are created by computing the support of the pattern with each restriction of the current border. Each infrequent restriction is replaced by its (more general) super restrictions in the lattice.

[1]It would also be possible to perform specialization steps top-down from the most general restrictions. Which approach is better has to be evaluated but is expected to depend on the domain.

```
predicate(uncovered(_,_)).
predicate(pass(_,_)).
predicate(closerToGoal(_,_)).

range(uncovered,[object, object]).
range(pass, [object, object]).
range(closerToGoal, [object, object]).

directSubClassOf(team1, object).
directSubClassOf(team2, object).
directInstanceOf(p6, team1).
directInstanceOf(p7, team1).
directInstanceOf(p8, team1).
directInstanceOf(p9, team1).
directInstanceOf(q6, team2).
directInstanceOf(q7, team2).
directInstanceOf(q8, team2).
directInstanceOf(q9, team2).

holds(uncovered(q6, q6),    12, 14).
holds(pass(p9, p8),         15, 17).
holds(closerToGoal(p8, p9), 11, 19).
holds(uncovered(p8, p8),    13, 21).
holds(closerToGoal(q8, q9), 16, 26).
holds(pass(p7, p6),         27, 29).
holds(closerToGoal(p6, p7), 23, 31).
holds(uncovered(p6, p6),    25, 33).
holds(uncovered(q9, q9),    30, 36).
holds(closerToGoal(q8, q6), 36, 40).
holds(pass(p9, p7),         39, 41).
holds(closerToGoal(p7, p9), 35, 43).
holds(uncovered(q8, q8),    42, 44).
holds(uncovered(p7, p7),    37, 45).
holds(pass(p8, p6),         51, 53).
holds(closerToGoal(q7, q6), 50, 54).
holds(closerToGoal(p6, p8), 47, 55).
holds(uncovered(p6, p6),    49, 57).
holds(pass(p8, p7),         65, 67).
holds(uncovered(q6, q6),    58, 68).
holds(closerToGoal(p7, p8), 61, 69).
holds(uncovered(p7, p7),    63, 71).
```

Figure 8: Sample input for the pattern mining algorithm

After all most special borders (that still lead to frequent patterns) are created for the single dimensions (step 1 in Fig. 7) the different borders have to be merged. While merging the class and variable restrictions all pairs of the elements of the most special borders of the class and variable restrictions are created. If any of the combinations leads to a non-frequent pattern a generalization step is performed using the restriction lattices. In this generalization step one of the restrictions is kept fixed while the other is replaced by more general restrictions in the generalization/specialization lattice until a frequent pattern was found again (in the worst case one of the restrictions would be finally replaced by the most general one in its lattice in order to still create a frequent pattern). After the first two borders were merged the process is repeated with the combined border and the remaining border of the temporal restrictions. This process is illustrated in Fig. 7.

4 Example

This section presents a sample run of the system. Fig. 8 presents the (manually created) input data which consists of some potential predicates of the soccer domain. The `predicate` and `range` entries give information about the predicates and their ranges to use while learn-

```
before(holds(_, _, E1),holds(_, S2, _)) :-
E1 < S2.

after(X,Y) :- before(Y,X).

equal(holds(_, S1, E1),holds(_, S2, E2)) :-
S1 == S2,
E1 == E2.

meets(holds(_, _, E1),holds(_, S2, _)) :-
E1 == S2.

isMetBy(X,Y) :- meets(Y,X).
...
younger(holds(_, _, E1),holds(_, S2, _)) :-
E1 < S2.
...
```

Figure 9: Examples for the logical represenations of temporal relations

ing. In this example there are just three predicates `uncovered`, `pass`, and `closerToGoal` and their ranges are all set to the most general class `object`. The `directSubClassOf` and `directInstanceOf` relations represent information of the class hierarchy (e.g., `team1` is a subclass of `object`) and assigns classes to objects (e.g., `p6` is an instance of the class `team1`). The following lines with the `holds` entries represent the validity intervals of certain predicates. The first line means that player `q6` is `uncovered` in the interval between timepoints 12 and 14[2].

Fig. 9 shows a snippet of different temporal relations which also have been given as input. The first example here says that an validity interval is `before` another one if its end time point is smaller than the start time point of the other. During the test run just a subset of the temporal relations introduced by Freksa (and some combinations) were used, namely *younger, older, younger & contemporary, older & contemporary,* and *head to head*.

The output of the pattern mining algorithm can be seen in Fig. 10 and Fig. 11. In the examples patterns of the size 2 and 3 were created. The output shows the created basic patterns and their support values and the merged restriction border, respectively. The class restrictions assign classes to the variables (in the sequence of their appearances). The variable restriction shows for each variable if it is unique (indicated by the value -1) or if it was unified with another variable (indicated by the index of the other variable, starting from index 1). The temporal restrictions represent the temporal relations between different predicates. Here, the first two numbers indicate which predicates are restricted by the temporal relation by showing the index (also starting from index 1).

The first restriction triple of the first basic patterns in Fig. 10 says that the first two variables are instances of the class `team1` and the remaining two are instances of `team2`, that the second variable was unified with the first one and the fourth with the third one, and the there is no temporal relation between the two predicates (`uncovered(_G29, _G30), uncovered(_G32, _G33)`). The support of this pattern (also with the restrictions) is 3. The first restriction triple of the second basic pattern includes a temporal restriction. It says that the validity interval of the first predicate (`pass`) is older and contemporary with the validity interval of the second predicate (`uncovered`). Some more learned patterns can be seen in Fig. 11 where the pattern size was set to 3.

[2]Please note that in the current version just binary relations are allowed, i.e., unary relations are represented by a twin entry of an identical object identifier.

```
BasicPattern: [uncovered(_G29, _G30),
               uncovered(_G32, _G33)]
Frequency: 3

  ClassRestr: [team1, team1, team2, team2]
  VarRestr:   varRestr(-1, 1, -1, 3)
  TempRestr:  [tempRestr(1, 2, none)]
  Freq:       3

  ClassRestr: [team2, team2, team1, team1]
  VarRestr:   varRestr(-1, 1, -1, 3)
  TempRestr:  [tempRestr(1, 2, none)]
  Freq:       3

BasicPattern: [pass(_G26, _G27),
               uncovered(_G29, _G30)]
Frequency: 5

  ClassRestr: [team1, team1, team1, team1]
  VarRestr:   varRestr(-1, -1, 2, 2)
  TempRestr:  [tempRestr(1, 2, olderContemp)]
  Freq:       5

  ClassRestr: [team1, team1, team2, team2]
  VarRestr:   varRestr(-1, -1, -1, 3)
  TempRestr:  [tempRestr(1, 2, older)]
  Freq:       3

BasicPattern: [closerToGoal(_G23, _G24),
               uncovered(_G29, _G30)]
Frequency: 5

  ClassRestr: [team1, team1, team1, team1]
  VarRestr:   varRestr(-1, -1, 1, 1)
  TempRestr:  [tempRestr(1, 2, none)]
  Freq:       5

  ClassRestr: [team1, team1, team2, team2]
  VarRestr:   varRestr(-1, -1, -1, 3)
  TempRestr:  [tempRestr(1, 2, none)]
  Freq:       3

BasicPattern: [closerToGoal(_G23, _G24),
               pass(_G26, _G27)]
Frequency: 5

  ClassRestr: [team1, team1, team1, team1]
  VarRestr:   varRestr(-1, -1, 2, 1)
  TempRestr:  [tempRestr(1, 2, youngerContemp)]
  Freq:       5

  ClassRestr: [team2, team2, team1, team1]
  VarRestr:   varRestr(-1, -1, -1, -1)
  TempRestr:  [tempRestr(1, 2, none)]
  Freq:       3
```

Figure 10: Examples of learned patterns with pattern size 2

5 Conclusion

In this paper we presented an approach to temporal pattern mining. One possible application of such learned patterns is the prediction of situations or behaviors by using (temporal) association rules. 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.

As the drawback of the approach is the high complexity of the mining algorithm as discussed in [Lattner and Herzog, 2004] we presented a way to reduce complexity in this paper. Instead of creating all combinations of the frequent restrictions of the different restriction kinds the algorithm just works with the most special (but still frequent) border of each single dimensions and still allows for extracting all frequent restrictions from the merged border by generalization. The approach has been implemented but not yet evaluated sufficiently, thus just a small sample run was presented in this paper.

Besides evaluation future work has to address further ways to reduce complexity, e.g., by developing heuristics which allow an efficient mining of patterns without cutting off a large number of potentially good patterns. In future work the performance of the learned patterns for predicting future behaviors and situations must also be analyzed.

```
BasicPattern: [pass(_G26, _G27),
               uncovered(_G28, _G29),
               uncovered(_G30, _G31)]
Frequency: 3

  ClassRestr: [team1, team1, team1, team1, team2, team2]
  VarRestr:   varRestr(-1, -1, 2, 2, -1, 5)
  TempRestr:  [tempRestr(1, 2, olderContemp),
               tempRestr(1, 3, none),
               tempRestr(2, 3, none)]
  Freq:       3

  ClassRestr: [team1, team1, team2, team2, team1, team1]
  VarRestr:   varRestr(-1, -1, -1, 3, 2, 2)
  TempRestr:  [tempRestr(1, 2, none),
               tempRestr(1, 3, olderContemp),
               tempRestr(2, 3, none)]
  Freq:       3

BasicPattern: [closerToGoal(_G23, _G24),
               pass(_G26, _G27),
               uncovered(_G28, _G29)]
Frequency: 5

  ClassRestr: [team1, team1, team1, team1, team1, team1]
  VarRestr:   varRestr(-1, -1, 2, 1, 1, 1)
  TempRestr:  [tempRestr(1, 2, youngerContemp),
               tempRestr(1, 3, none),
               tempRestr(2, 3, olderContemp)]
  Freq:       5

  ClassRestr: [team1, team1, team1, team1, team2, team2]
  VarRestr:   varRestr(-1, -1, 2, 1, -1, 5)
  TempRestr:  [tempRestr(1, 2, youngerContemp),
               tempRestr(1, 3, none),
               tempRestr(2, 3, none)]
  Freq:       3

  ClassRestr: [object, object, team1, team1, team2, team2]
  VarRestr:   varRestr(-1, -1, -1, -1, -1, 5)
  TempRestr:  [tempRestr(1, 2, none),
               tempRestr(1, 3, none),
               tempRestr(2, 3, older)]
  Freq:       3
```

Figure 11: Examples of learned patterns with pattern size 3

References

[Agrawal and Srikant, 1994] Rakesh Agrawal and Ramakrishnan 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.

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imielinski, and Arun 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.

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Dehaspe and Raedt, 1997] Luc Dehaspe and Luc De Raedt. Mining association rules in multiple relations. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 125–132. Springer-Verlag, 1997.

[Dehaspe and Toivonen, 2001] Luc Dehaspe and Hannu Toivonen. Discovery of relational association rules. In *Relational Data Mining*, pages 189 – 208. Springer-Verlag New York, Inc., 2001.

[Freksa, 1992] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227, 1992.

[Höppner, 2001] Frank 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.

[Huang *et al.*, 2003] Zhanxiang Huang, Yang Yang, and Xiaoping Chen. An approach to plan recognition and retrieval for multi-agent systems. In Mikhail Prokopenko, editor, *Workshop on Adaptability in Multi-Agent Systems, First RoboCup Australian Open 2003 (AORC-2003)*, Sydney, Australia, 2003. CSIRO.

[Kaminka *et al.*, 2003] Gal Kaminka, Mehmet Fidanboylu, Allen Chang, and Manuela Veloso. Learning the sequential coordinated behavior of teams from observation. In Gal Kaminka, Pedro Lima, and Raul Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI, LNAI 2752*, pages 111–125, Fukuoka, Japan, 2003.

[Koperski and Han, 1995] Krzysztof Koperski and Jiawei 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.

[Lattner and Herzog, 2004] Andreas D. Lattner and Otthein 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.

[Malerba and Lisi, 2001] Donato Malerba and Francesca 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.

[Mannila *et al.*, 1997] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[Mennis and Liu, 2003] Jeremy Mennis and Junwei 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.

[Miene *et al.*, 2004a] Andrea Miene, Andreas D. Lattner, Ubbo Visser, and Otthein 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.

[Miene *et al.*, 2004b] Andrea Miene, Ubbo Visser, and Otthein 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.