Temporal Pattern Mining in Logistics

Andreas D. Lattner, Tjorben Bogon, René Schumann, Ingo J. Timm

Johann Wolfgang Goethe-Universität Frankfurt,
Institute of Computer Science, Information Systems and Simulation
P.O. Box 11 19 32, D-60054 Frankfurt am Main, Germany
[lattner|tbogon|reschu|timm]@cs.uni-frankfurt.de

ABSTRACT
Modern technologies like RFID, GPS, and wireless networks provide means for an automated tracking of goods, containers, and transportation vehicles. Collecting data about positions and movements of different actors and objects is a prerequisite for an automated analysis of ongoing logistic processes. As extensive information about objects and their relations is available, it is possible to apply data mining techniques in order to identify patterns in the data. In this work, we analyze the requirements for mining patterns in the logistics domain and present an algorithm for mining temporal patterns and prediction rules from complex representations of scenes. An implementation of the mining approach is applied to two test scenarios in order to create and test prediction rules.

INTRODUCTION
In logistics scenarios, there exist many different kinds of objects like transport vehicles (e.g., trucks, ships, or planes), different actors or organizations (e.g., depots, carriers, manufacturers), highways or tracks, transshipment stations, etc. Different events can occur in the dynamic environment like traffic jams, weather events, break downs of transport vehicles, or delivery delays of some goods. It would be very valuable to identify repeating patterns that lead to certain situations in order to predict, for instance, that for the given situation it is likely that rescheduling will be necessary later on. Having this information it would be possible to initiate some counter-actions earlier in order to avoid financial loss or penalty payments. Such a pattern could be, for instance: If the capacity load of depot X is high and transportation vehicles Y on the way to X has a break-down, it is likely that some goods Z have to be transferred to some other depot.

Modern technologies like RFID, GPS, and wireless networks provide means for an automated tracking of goods, containers, and transportation vehicles. Having available data about positions and movements of different objects enables an automated analysis of ongoing logistic processes. If extensive information about objects and their relations is available, it is possible to apply data mining techniques in order to identify patterns in the data. Patterns can be transformed to prediction rules by estimation of causal interdependencies. If it was known that certain situations lead to some future events or situations with some probability, this information could be used to improve the decision making process in logistic scenarios, for instance by avoiding critical situations. The logistics domain features high complexity w.r.t. different objects and relations appearing in scenes on the one hand and demands means for the representation of temporal information on the other hand.

The requirements for learning such patterns are manifold. On the one hand, the complex situations need a sophisticated representation formalism in order to capture the scenes and to represent the patterns. On the other hand, the patterns should be comprehensible and easily applicable to future situations. The requirements are:

- The scene representation has to provide means to represent objects, their properties, as well as interrelations among objects.
- Due to the dynamics, an explicit representation of the temporal dimension is needed. As different relations and events can exist concurrently, it is required that the representation can also deal with such concurrent situations.
- Capturing conceptual information about object's classes and possible interrelations is needed to guide the search for patterns.
- Background knowledge is necessary in order to cover general rules of the domain without demanding the user to provide (redundant) knowledge about each individual object (truck, depot etc.).
- Unsupervised identification of generic frequent patterns, i.e., abstracting from the concrete objects and identifying common patterns.
- Generation of prediction rules to be applied to future situations.
- Pattern matching methods in order to detect patterns (or preconditions of prediction rules).
- Evaluation of created prediction rules.

The paper is structured as follows: Section 2 presents related work, addressing learning in logistics and pattern mining approaches. Our pattern mining approach is described in section 3. In section 4, we give an example scenario and show created patterns before conclusions are drawn in section 5.

RELATED WORK

In logistics scenarios usually a huge amounts of data are involved. Different learning approaches have been used in order to handle this data and to improve the solutions of logistics problems. For instance, neural networks are used to optimize traditional logistics problems like the Travelling Salesman Problem, routing problems, bankruptcy prediction, and dispatching problems (Wilppu, 1999). Another task in logistics is to manage and control a supply chain. With the rising connectivity of the world (airplains, ships) the supply chains spread wide over all continents. Bruzzone and Orsoni describe three different approaches how costs in supply chains could be reduced using techniques from the areas of artificial intelligence, stochastic and mathematics (Bruzzone and Orsoni, 2003). Another approach to handle the complexity of global supply chains is described in (Pontrandolfo et al., 2002). Pontrandolfo et al. use Reinforcement Learning to let different sites learn to work efficiently together. Every site is represented by an agent who acts with semi-Markov decision processes. Traffic route planning problems can also be described with a multi-agent system. Gehrke and Wojtusiak present an approach to react online on influences from the environment (for example weather) (Gehrke and Wojtusiak, 2008a, b). The approach tries to identify the best routes by taking into account the wetness and the speed limits of the roads. Every truck is represented by an agent and can dynamically react to new events. The (propositional) rule induction system AQ21 has been used to set up prediction rules.

In the recent years, different learning approaches have been presented which satisfy the requirements described above partially. In the field of Inductive Logic Programming (ILP), various approaches have been presented that can deal with relational representations. ILP approaches like FOIL and Progol (Muggleton, 1995; Quinlan, 1990) are supervised and thus do not identify frequent patterns from data as desired. The rule learner WARMR combines ideas from the fields of association rule mining and ILP (Dehaspe and Toivonen, 1999) but does not provide means for explicit representation of the temporal dimension. Different approaches to sequential or temporal pattern mining can be found, e.g., (Agrawal and Srikant, 1995; Höppner, 2003; Mannila et al., 1997). But most of these approaches cannot deal with relational representations as required in our case.

Jacobs and Blockeel apply the ILP association rule learner WARMR in order to mine shell scripts from Unix command shell logs (Jacobs and Blockeel, 2001). Although having a complete different domain, the resulting patterns can represent complex interrelations and temporal relations (sequences). Log files of shells can be seen as a sequence of commands. Frequent patterns from such command sequences can be interpreted as shell scripts. The challenge is to deal with the arguments in commands and use variables in patterns in order to represent that the same argument (e.g., a file name) should be used by a different command. Jacobs and Blockeel use WARMR for the generation of scripts and also present some methods for speedup by splitting up the learning task and using the so called minimal occurrence algorithm. Command sequences are represented by a stub relation with unique identifier, execution time, and command (e.g., stub(1,2008,'cp')) and parameter relations (e.g., parameter(1,1,'file1') and parameter(1,2,'file2')). The learning task is split up by first finding the frequent sequences of commands and then taking the parameter information into account. The minimal occurrence algorithm takes into account the sequential information in the query generation process and can thus prune some of the patterns that cannot be frequent any more. It also utilizes the identifiers at which sequences of the previous step start for the calculation of occurrences of a new sequence (Jacobs and Blockeel, 2001).

Masson and Jacquenet address the mining of frequent logical sequences (Masson and Jacquenet, 2003). They extend the SPIRIT system (Garofalakis et al., 1999) to discover logical sequences and introduce the SPIRIT-LOG algorithms. The major adaptations w.r.t. SPIRIT are done in the generation and pruning functions. Candidate generation is extended to handle logical sequences with variables and in the pruning step an inclusion test between logical sequences (including unification of variables) are developed. SPIRIT-LOG can only create patterns of contiguous predicates, i.e., no gaps are allowed in the sequential patterns. Furthermore, it is not possible to use background knowledge in the form of clauses (cf. (Lee and De Raedt, 2004)).

Lee and De Raedt introduce the logical language SeqLog for the representation of sequential logical data (Lee, 2006; Lee and De Raedt, 2004). The sequence itself is represented as a sequence of logical atoms. Additionally, it is possible to specify background knowledge as DATALOG style clauses. The mined patterns consist of a sequence of logical atoms. Two adjacent atoms in the pattern can be specified as direct (temporal) neighbors or allow for having other elements between them (denoted by the < symbol). Lee and De Raedt introduce the mining system MineSeqLog which mines the borders of the solution space for an input sequence and a conjunction of monotonic and anti-monotonic constraints on the patterns (Lee and De Raedt, 2004). However, this mining approach cannot mine any patterns with concurrent occurrences of events or activities.

MINING TEMPORAL PATTERNS

The representation of scenes used here is based on Allen's theory of action and time (Allen, 1984); it is a set of time intervals representing different events and relations between objects in the scene with a temporal validity interval. An example for such a representation is shown in Fig. 1. The temporal dimension is shown on the x axis. The intervals (like "capacity_load(s1, high)") determine the validity of certain relations or activities. Formally, there exists a start and end time for each interval. As it can be seen in Fig. 1, there are various intervals "active" concurrently.

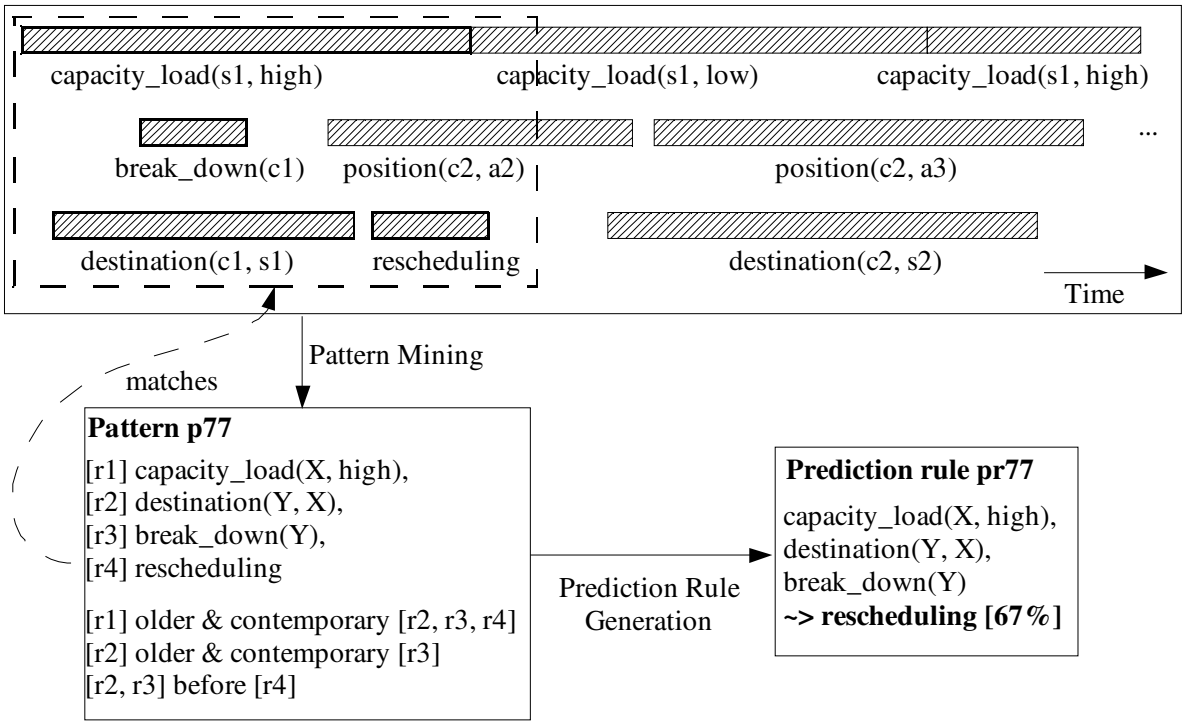

Figure 1: Pattern Mining and Prediction Rule Generation

We define a temporal pattern as a set of predicates with temporal restrictions (interrelations) and concept restrictions. Predicates can represent relations between objects (e.g., "capacity_load(depot1, high)") or more abstract statements with variables (e.g., "break-down(X)"). The temporal restriction describes temporal constraints between predicate (time interval) pairs using interval relations as they have been introduced by (Allen, 1983) and (Freksa, 1992) (e.g., high capacity load in depot Y happens before break-down of Y). As a third aspect, variables can be restricted to certain concepts (e.g., "X must be an instance of Truck"). A temporal pattern matches to a situation if all variables can be unified with objects in the situation without violation of the temporal or conceptual constraints.

The mining algorithm itself can be seen as an extension of the well known association rule mining algorithm Apriori (Agrawal and Srikant, 1994), additionally inspired by different approaches to relational as well as sequential association rule mining (Dehaspe and Toivonen, 1999; Höppner, 2003; Lee, 2006). It performs a top-down search through the pattern space starting with the most general pattern. The most general pattern consists of no predicate and thus matches any situation. In different refinement steps the patterns are specialized by five refinement operators:

- Lengthening: Extend the pattern with a predicate.
- Variable unification: Unify two variables X and Y, i.e., X=Y.
- Instantiation: Replace a variable by one of the instances of the corresponding concept.
- Temporal refinement: Restrict the set of allowed temporal relations among the predicates.
- Concept refinement: Specialize the concept of one of the variables by replacing it with one of its sub concepts (e.g., from "vehicle" to "truck").

The search is restricted by a support threshold (as it is the case in association rule mining; (Agrawal and Srikant, 1994)). If a pattern does not exceed the support threshold, all its refinements also cannot exceed this threshold. The reason for this is that the refinements are a further restriction and cannot lead to more matches.

For pattern matching, a sliding window is used. A pattern matches at a window position if there exists a variable assignment such that all predicates are matched, and no concept or temporal restriction is violated. It is possible to use different sets of (time) interval relations, e.g., those defined by (Allen, 1983).

In a second step, the generated patterns are transformed into prediction rules. This is done by splitting the pattern into two parts: The precondition and consequence part. For each prediction rule, a confidence value for the rule is computed by counting the situations where the preconditions and consequence occur together. If such a prediction rule is applied to future situations, it is checked if the precondition matches a scene. If this is the case, the consequence part is provided as prediction (with the assigned confidence value). A time interval representation as well as the mining process is illustrated in Fig. 1. Further details about the mining algorithm can be found in (Lattner, 2007).

SCENARIO

The approach presented in the previous section has been applied to the test scenario sketched in Fig. 2. This scenario and the upcoming test scenario have been created manually for illustration purposes. The first part of the input file (left-hand side in Fig. 2) describes the used parameter for learning. The size of the sliding window (for pattern matching) is set to 12, the minimal frequency is 0.2, and the minimal and maximal pattern sizes are 1 and 4, respectively. The maximal refinement level (number of pattern refinement steps) is set to 20. The sliding window size determines what time window is actually taken into account for pattern matching. The pattern sizes restrict how many predicates can be added to a temporal pattern. The frequency threshold indicates in how many situations a pattern must occur in order to be regarded as "frequent". The predicate templates section in the input file defines the predicates (and the valid concepts of their parameters) which can be used during mining. The predicate template "destination(vehicle, depot)", for instance, defines that the destination relation can exist among vehicles and depots. Concept and instance information is provided for setting up the concept hierarchy (is-a relation among concepts) and the instance-of relations between objects and concepts. In this scene description, there are definitions of the concepts depot, vehicle, location, and tour (all direct sub concepts of object; the formalism also allows deeper concept hierarchies). The objects v1, v2 etc. are instances of the concept "vehicle".

The actual dynamic scene description is shown on the right-hand side of Fig. 2. The set of holds predicates describe which predicates are valid at what times by providing the start and end times for each relation. The first line represents the fact that there has been high capacity load in depot d1 during the time interval from 5 to 20. This is an alternative representation for the time line as illustrated in Fig. 1. The dynamic scene consists of some repeating combinations of predicates.

An implementation of the pattern mining approach (MiTemP; implemented in XSB Prolog (Sagonas et al., 2006)) has been applied to the scenario in order to create frequent patterns and prediction rules. Two examples for prediction rules are shown on the left of Fig. 3. The first prediction rule can be read like this: If there is a precondition with a depot A having high capacity load, a vehicle B with destination A, and a breakdown of B, then it is very likely (for this example even a likelihood of 1.0) that a rescheduling will be necessary later on. There is a set of temporal relations among the different predicates stating that high_capacity_load(A) must be older and contemporary with break_down(B) etc. This prediction rule (no. 1) is then applied to another dynamic scene example shown on the right-hand side of Fig. 3 in order to test if a general rule has been identified which also applied to situations with other vehicles and depots.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters
mitempParam(windowSize, 12).
mitempParam(minFrequency, 0.2).
mitempParam(minPatternSize, 1).
mitempParam(maxPatternSize, 4).
mitempParam(maxLevel, 20).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Predicate templates

predicate(high_capacity_load(depot)).
predicate(destination(vehicle,depot)).
predicate(break_down(vehicle)).
predicate(position(vehicle,location)).
predicate(rescheduling(tour)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Concept and instance information

isA(depot, object).
isA(vehicle, object).
isA(location, object).
isA(tour, object).

directInstanceOf(v1, vehicle).
directInstanceOf(v2, vehicle).
directInstanceOf(v3, vehicle).
directInstanceOf(v4, vehicle).
directInstanceOf(v5, vehicle).
directInstanceOf(a1, location).
directInstanceOf(a2, location).
directInstanceOf(a3, location).
directInstanceOf(d1, depot).
directInstanceOf(d2, depot).
directInstanceOf(d3, depot).
directInstanceOf(t, tour).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dynamic scene (pattern mining)

holds(high_capacity_load(d1), 5, 20).
holds(destination(v1,d1), 6, 15).
holds(break_down(v1), 8, 14).
holds(rescheduling(t), 16, 19).
holds(position(v2, a1), 10, 20).
holds(position(v3, a3), 8, 18).

holds(high_capacity_load(d1), 45, 60).
holds(destination(v3,d1), 46, 55).
holds(break_down(v3), 48, 54).
holds(rescheduling(t), 56, 59).
holds(position(v5, a1), 50, 60).
holds(capacity_load(d3,high), 48, 58).

holds(high_capacity_load(d2),85, 100).
holds(destination(v5,d2), 86, 95).
holds(break_down(v5), 88, 94).
holds(rescheduling(t), 96, 99).
holds(break_down(v1), 90, 100).
holds(position(v1, a2), 88, 98).

holds(high_capacity_load(d2),125,140).
holds(destination(v2,d2), 126, 135).
holds(break_down(v2), 128, 134).
holds(rescheduling(t), 136, 139).
holds(capacity_load(d1,low),130, 140).
holds(position(v3, a2), 128, 138).

holds(high_capacity_load(d3),165,180).
holds(destination(v4,d3), 166, 175).
holds(break_down(v4), 168, 174).
holds(rescheduling(t), 176, 179).
holds(break_down(v5), 170, 180).
holds(position(v2, a3), 168, 178).
```

Figure 2: Pattern mining input

Testing the learned prediction rules on the new scene leads to a set of time intervals where the precondition of the prediction rule is actually satisfied, for instance in the time interval from 8-27 (Fig. 3, bottom left). Within this interval all three predicates are found within a sliding window in the scene and thus it is predicted that a rescheduling is likely to follow. In this test scenario, for all five occurrences of the pattern, the prediction rule has correctly identified the upcoming rescheduling event.

CONCLUSION

In this paper, we have proposed an approach to temporal pattern mining in logistics. The motivation for the identification of patterns is to set up (temporal) prediction rules which might identify critical situations before they occur. The domain of logistics demands a high representational expressiveness due to various types of objects, relations, and the dynamic aspects of scenes. The proposed mining approach can identify frequent pattern from a multi-relational and temporal representation of scenes. In the first step of the mining algorithm, frequent patterns are identified. This is done by a generic-to-specific search starting from the most general pattern and applying different refinement operations. The second step splits the patterns in precondition and consequence parts and computes the confidence of the pattern (i.e., the fraction of occurrences where precondition and consequence appear together).

For illustration, the learning approach has been used to learn prediction rules from a training scenario. One of the created prediction rules has been applied to a test scenario showing that a general prediction rule (independent from concrete instances) could be found. Nevertheless, this is only the first step towards an evaluation of the approach in the logistics domain. For future work, it

would be interesting to apply the mining approach to some real-world data and to find out, how identified prediction rules perform on unseen situations.

```
...
===== Prediction rule 1:
[high_capacity_load(A),
destination(B,A),  break_down(B)]  =>
[rescheduling(t)]
temp(tr([olderContemp],[olderContemp],
[olderContemp]),tr([olderContemp],
[before]), tr([before]))
conceptRestr(depot,  vehicle,  depot,
vehicle, tour)

Eval: 0.6779
(f: 0.2674, c: 1.0000, j: 0.1497,
 s: 1.0000, r: 0.6500, p: 1.0000)
===== Prediction rule 2:
[high_capacity_load(A),
destination(B,A)]  =>  [break_down(B),
rescheduling(C)]
temp(tr([olderContemp],[olderContemp],
[olderContemp]),tr([olderContemp],
[before]), tr([before]))
conceptRestr(depot,  vehicle,  depot,
vehicle, tour)

Eval: 0.6805
(f: 0.2674, c: 0.9524, j: 0.2631,
 s: 1.0000, r: 0.6000, p: 1.0000)
=====
...


Application of prediction rule 1:

Precondition matches at:
[i(8,27), i(49,66), i(88,107),
 i(129,147), i(169,187)]
Quality => 1.0000
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dynamic scene (testing)

holds(high_capacity_load(d6), 4, 21).
holds(destination(v6,d6), 5, 16).
holds(break_down(v6), 8, 15).
holds(rescheduling(t), 17, 20).
holds(position(v10, a5), 13, 16).
holds(position(v8, a6), 14, 19).

holds(high_capacity_load(d6), 43, 62).
holds(destination(v8,d6), 44, 56).
holds(break_down(v8), 49, 54).
holds(rescheduling(t), 57, 60).
holds(position(v7, a5), 53, 61).
holds(capacity_load(d5,high), 46, 55).

holds(high_capacity_load(d4),84, 101).
holds(destination(v10,d4), 87, 96).
holds(break_down(v10), 88, 95).
holds(rescheduling(t), 97, 100).
holds(break_down(v6), 88, 98).
holds(position(v6, a4), 87, 96).

holds(high_capacity_load(d4),123,142).
holds(destination(v7,d4), 127, 136).
holds(break_down(v7), 129, 135).
holds(rescheduling(t), 138, 140).
holds(capacity_load(d6,low),129, 135).
holds(position(v8, a4), 127, 134).

holds(high_capacity_load(d5),163,181).
holds(destination(v9,d5), 166, 177).
holds(break_down(v9), 169, 175).
holds(rescheduling(t), 178, 179).
holds(break_down(v10), 174, 183).
holds(position(v7, a6), 167, 179).
```

Figure 3: Prediction rules and test scene

REFERENCES

Agrawal, R. and Srikant, R. (1994). "Fast Algorithms for Mining Association Rules." Proceedings of the 20th International Conference on Very Large Data Bases (VLDB) pp. 487-499).

Agrawal, R. and Srikant, R. (1995). "Mining Sequential Patterns." Proceedings of the IEEE International Conference on Data Engineering pp. 3-14). Taipei, Taiwan.

Allen, J. F. (1983). "Maintaining Knowledge about Temporal Intervals." Communications of the ACM 26(11): 832-843.

Allen, J. F. (1984). "Towards a General Theory of Action and Time." Artificial Intelligence 23(2): 123-154.

Bruzzone, A. G. and Orsoni, A. (2003). "AI and Simulation-Based Techniques for the Assessment of Supply Chain Logistic Performance." Proceedings 36th Annual Simulation Symposium (ANSS-36 2003) pp. 154-164). Orlando, Florida: IEEE Computer Society.

Dehaspe, L. and Toivonen, H. (1999). "Discovery of frequent DATALOG patterns." Data Mining and Knowledge Discovery 3(1): 7 - 36.

Freksa, C. (1992). "Temporal Reasoning Based on Semi-Intervals." Artificial Intelligence 54(1-2): 199-227.

Garofalakis, M. N., Rastogi, R. and Shim, K. (1999). "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints." Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99) pp. 223-234). Edinburgh, Scotland, UK.

Gehrke, J. D. and Wojtusiak, J. (2008a). A Natural Induction Approach to Traffic Prediction for Autonomous Agent-based Vehicle Route Planning Report No. 08-3: MLI, George Mason University, Fairfax, VA, USA.

Gehrke, J. D. and Wojtusiak, J. (2008b). "Traffic Prediction for Agent Route Planning." International Conference on Computational Science 2008. Krakow, Poland: Springer-Verlag.
Höppner, F. (2003). Knowledge Discovery from Sequential Data, Technische Universität Braunschweig.
Jacobs, N. and Blockeel, H. (2001). "From Shell Logs to Shell Scripts." In C. Rouveirol and M. Sebag (eds.), ILP 2001 pp. 80-90): Springer-Verlag Berlin Heidelberg.
Lattner, A. D. (2007). Temporal Pattern Mining in Dynamic Environments: Akademische Verlagsgesellschaft Aka GmbH Berlin.
Lee, S. D. (2006). Constrained Mining of Patterns in Large Databases, Albert-Ludwigs-Universität Freiburg.
Lee, S. D. and De Raedt, L. (2004). "Constraint Based Mining of First Order Sequences in SeqLog." In R. Meo, P. L. Lanzi and M. Klemettinen (eds.), Database Support for Data Mining Applications pp. 154-173): Springer Berlin / Heidelberg.
Mannila, H., Toivonen, H. and Verkamo, A. I. (1997). "Discovery of Frequent Episodes in Event Sequences." Data Mining and Knowledge Discovery 1: 259-289.
Masson, C. and Jacquenet, F. (2003). "Mining Frequent Logical Sequences with SPIRIT-LOG." In S. Matwin and C. Sammut (eds.), ILP 2002 pp. 166-182): Springer-Verlag Berlin Heidelberg.
Muggleton, S. (1995). "Inverse Entailment and Progol." New Generation Computing 13(3-4): 245-286.
Pontrandolfo, P., Gosavi, A., Okogbaa, O. G. and Das, T. K. (2002). "Global supply chain management: a reinforcement learning approach." International Journal of Production Research 40(6): 1266-1317.
Quinlan, J. R. (1990). "Learning Logical Definitions from Relations." Machine Learning 5: 239-266.
Sagonas, K., Swift, T., Warren, D. S., Freire, J., Rao, P., Cui, B., Johnson, E., Castro, L. d., Marques, R. F., Dawson, S. and Kifer, M. (2006). "The XSB System Version 3.0 - Volume 1: Programmer's Manual, Volume 2: Libraries, Interfaces, and Packages."
Wilppu, E. (1999). Neural Networks and Logistics Report No. TUCS-TR-311: Turku Centre for Computer Science, Turku School of Economics and Business Administration, Finland.