Finding Temporal Patterns using Constraints on (Partial) Absence, Presence and Duration

S. Peter and F. Höppner

Ostfalia University of Applied Sciences
Robert Koch Platz 10-14, D-38440 Wolfsburg

Abstract. When the evolution of variables over time is relevant to a classification task, established classifiers cannot be applied directly as the typical input format (data table) is not appropriate. We propose a new representation of temporal patterns that includes constraints on (partial) presence, (partial) absence as well as the duration of temporal predicates. A general-to-specific search-based algorithm is presented to derive classification rules. The approach evaluates promising on artificial and real data.

1 Introduction

One important aspect of data mining is to identify dependencies and interrelationships that were unknown to the user before and deliver them in an easily understandable representation. Rule-based systems and decision trees, which fulfill such requirements, assume a databases of cases with characterizing features that held at the time of recording the case. In many areas, however, a *case* stretches over time, such as the traffic density over one day, the medication of a patient over the duration of illness, control variables of a production process over a production cycle, a workflow of a business process, etc. In such domains it is often not feasible to drop the temporal information, sometimes even the order of isolated events is not sufficient, but their temporal extent and contemporaneity is important. The discovery of complex relationships of the latter kind without compromising their interpretability requires a descriptive, graphical representation of the discovered patterns. In this paper, we propose a new temporal pattern representation and suggest an algorithm to learn classification rules automatically from temporal data.

2 Representing Evolving Data

We consider the history of a binary attribute as a temporal predicate. Denoting the temporal dimension by $\mathbb{T}$, a temporal predicate P_l is a function $P : \mathbb{T} \to \mathbb{B}$. l is called the label of the predicate P. We assume that all available data may be represented by such temporal predicates. The development of a nominal attribute with domain $\{u, v, w\}$ may be represented by three predicates P_u, P_v and P_w, denoting when each of the values held. A series of numerical values (time series) can be represented by extracting different predicates such as $P_{\text{increasing}}$ or $P_{\text{high-valued}}$. A set of such predicates (which we will call history H) is often depicted by listing the various predicates against the temporal dimension (cf. Fig. 1). A suitable data representation is a (labelled) sequence of temporal intervals that indicate when the predicate did hold. Such a representation is used frequently, e.g. in the medical domain [7].

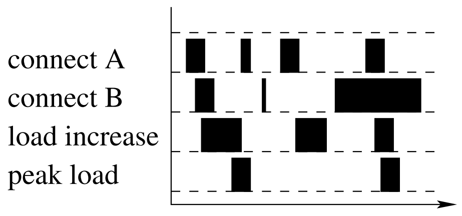


Fig. 1. Representation of Evolving Data: the black rectangles denote the intervals when the predicate (labels to the left) holds.

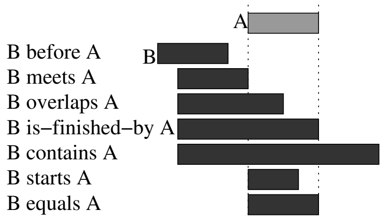


Fig. 2. Thirteen possible relationships between two intervals. The inverse relationships (after ↔ before) have been omitted.

In the context of classification tasks the goal is to describe circumstances (prototypical histories) under which a certain target variable is likely to occur. Note that in contrast to stream mining approaches, where a single but potentially infinite stream of data is considered, we assume that multiple finite, labelled histories are available. Various ways to define such patterns in a stream of labeled intervals have been proposed in the literature, many of them relying on Allen's interval relationships [1] (cf. Fig. 2) or variants thereof. Some approaches define a history by specifying the exact relationship for every pair of intervals [3], others allow for a set of possible relationships [4]. The representation by sequences of chords [6] uses a partially ordered sequence of simultaneous (sub-) intervals to define a pattern. Some other proposals consider a different set of interval relationships or specify the next relationship only with respect to the union of the pattern discovered so far [5].

While these approaches have their individual strengths, they also have their weaknesses when it comes to represent certain simple situations. Thinking of predicting a certain state of some network server (breakdown, overload, attack, malfunction, etc.) on the history of, say, the last 24 hours, a situation as simple as "there was only one connection to server A" (during the last n hours) is usually impossible to discover for the approaches based on association rules [3, 6], as they count occurrences of events only and rely on a quickly decreasing count of co-occurrences, such that an inclusion of *absent features* during counting undermines the initial assumptions of association mining. A situation like "there was a connection to B while the connection to A was lost" is impossible to represent for the approaches that rely on explicitly given interval relationship, as the exact position of B relative to A is not known [3]. Temporal constraints "the connection to A was lost for at least 4 hours" or "... at most 4 hours" are usually ignored completely or introduced in a postprocessing step.

We propose a new notion of a pattern, called *template history*, that shall be matched against an existing history later. To maximally support the understandability of the template, we keep the basic representation of Fig. 1, but relax the temporal alignment to allow for successful matching despite of dilational and translational effects. We do not care about a 1:1 mapping of time points (as in dynamic time warping) but concentrate on a few relevant points in time that are indicated by vertical *alignment lines*. This corresponds roughly to a discretization of the temporal axis, but we do not finally fix the position of the lines but adjust them at match-time. If m labels/predicates and $n+1$ adjustment lines are given, we obtain an $m \times n$ matrix which allows us to pose different conditions on the predicates in each of the matrix cells. We distinguish four different conditions:

Definition 1 (predicate constraint). *Give a temporal interval* $T \subseteq \mathbb{T}$ *and a predicate* P*, we say (a)* P is present during T *if* $\forall t \in T : P(t)$*, (b)* P is absent during T *if* $\forall t \in T : \neg P(t)$*, (c)* P exists during T *if* $\exists t \in T : P(t)$ *and (d)* P disappears during T *if* $\exists t \in T : \neg P(t)$*. If no condition is posed, we say* P is unconstrained during T*. By* $\mathbb{C}$ *we denote the set of constraints { present, absent, exists, disappears, unconstrained }.*

Every cell of the template matrix is now filled with one of the five constraints. Additionally a template may contain temporal constraints on the distance between the alignment lines (block duration).

Definition 2 (template). *A tuple* $T = (L, n, C, D)$ *is called a template if* L *is a set of labels,* $n \in \mathbb{N}$*,* $C : L \times \{1, .., n\} \to \mathbb{C}$ *and* $D : \{1, .., n\} \to (\mathbb{T} \cup \{\infty\})^2$ *with* $1 \leq D(i)_0 \leq D(i)_1$ *for all* $1 \leq i \leq n$*.*

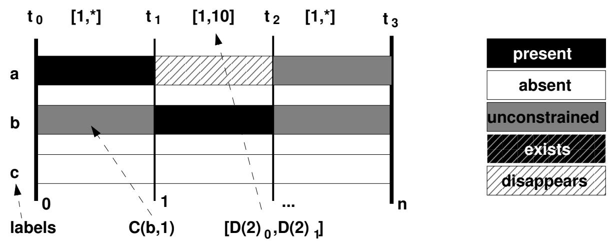


Fig. 3. Illustration of the template definition. The map D defines the block durations (time interval between alignment lines) and is shown on the top. The predicate constraints are coded by color.

Figure 3 shows an example template with $n = 3$ blocks and thus four vertical alignment lines, where the leftmost and rightmost alignment line shall always represent the start and end of the history. The bottom row declares that a predicate P_c is absent in the whole history. Somewhere in the history (2nd block), P_b is present (P_b may be present or not elsewhere). P_a is present from the very beginning, but disappears while P_c is present in the 2nd block. The duration of the first block is arbitrary, the duration of the second block lies within $[1, 10] = [D(2)_0, D(2)_1]$ time units, the last block may again have any (positive) duration.

Matching a template to a real history involves two steps: Firstly, the alignment lines need to be positioned appropriately such that, secondly, all temporal and predicate constraints hold.

Definition 3 (match). *Let $T = (L, n, C, D)$ be a template and H be a history. Let $[t_{\min}, t_{\max}]$ be the smallest interval subsuming $\cup_{P\in H} dom(P)$. T matches a history H if and only if (a) there is a predicate $P_l \in H$ for every $l \in L$, (b) there are $t_i \in \mathbb{T}$, $0 \leq i \leq n$, with $t_0 = t_{\min}$, $t_i \leq t_{i+1}$, $t_n = t_{\max}$, (c) for every $l \in L$ and $i \in \{1, .., n\}$ the constraint $C(l, i)$ holds for P_l within $[t_i, t_{i+1})$ and finally (d) for all $1 \leq i \leq n$: $t_i - t_{i-1} \in \Delta_i$ with $\Delta_i = [D(i)_0, D(i)_1]$.*

3 Finding Patterns

Next, we propose a method to explore the space of templates to discriminate differently labelled histories. The search algorithm implements a general-to-specific search: It begins with a pattern, which matches all instances, and tries to specialize it further to improve some measure of interestingness (we use the J-measure [8] as it balances the generality (applicability of the rule) and the interestingness (deviation from a priori knowledge)). An initial template that matches all histories must have at least one block, all matrix constraints are *unconstrained* and so are the temporal constraints $[1, \infty]$. While a propositional rule can only be specialized by an additional condition (like *outlook=sunny*), there are at least three ways to specialize a template: we may look at it in a finer resolution (by adding another alignment line), we may change or add a predicate constraint (for some label and block), or may introduce or change an existing temporal constraint. We thus have chosen three different specialization operators to address each of these aspects.

The general idea for all refinement operators is to search for specializations that improve the measure of interestingness, which basically requires that the specialized pattern still matches the positive instances but less negatives.

Adding a predicate constraint. Suppose we want to specialize the pattern in Fig. 3 by an additional constraint for label X (which is *unconstrained* so far). If four instances are given that match this template, we have to inspect the occurrences of X relative to the alignment lines of the template. This is shown (only for variable X and the four cases) in Fig. 4. We create confusion matrices for all possible refinements (each block and constraint combination) of the predicate X by counting how the instances will be classified by the considered specialization. For instance, the specialization shown in Fig. 5 would perfectly discriminate between the classes, as P_X is present in the first block only for

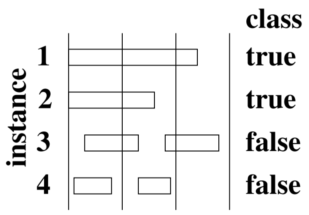


Fig. 4. Relative occurrence of the predicate X to the matches of the pattern shown in Fig. 3 in the instances

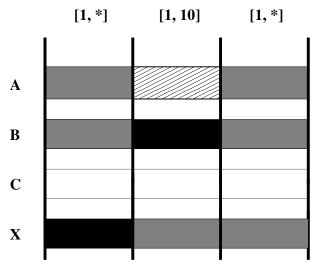


Fig. 5. Refined pattern for classify the instances correctly

the positive examples. Finally we apply the measure of interestingness on all confusion matrices to find the best refinement for X. We instantiate such a constraint refinement for every label to pick the best constraint specialization.

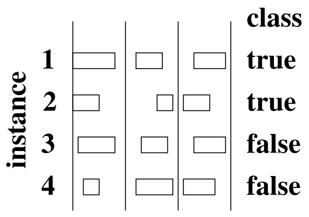


Fig. 6. Relative occurrence of the predicate X to the matches of the pattern shown in Fig. 3 in the instances

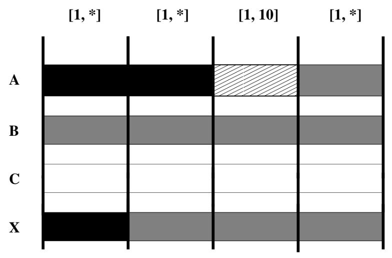


Fig. 7. Refined pattern for classify the instances correctly

Adding an alignment line. The second operator adds new alignment lines to the template. Similar to Fig. 4, a different constellation of cases is shown in Fig. 6. Apparently an *exists*-constraint would match the positive classes (in any of the blocks), but unfortunately, it would also match the negative cases. However, we notice that all the positive cases start at the beginning of the first block, whereas the negative cases do not. By introducing a new alignment line that subdivides the first block, we may pose a *present*-constraint on the left part of the first block. The original template from Fig. 3 is extended by a new vertical line as shown in Fig.7. As before, we again determine confusion matrices for different specializations and evaluate them for each block and combination of *absent*/*present*-constraints.

Adding a temporal constraint. Finally, the third operator tries to find a block length for a specific block, so that the new block length mostly holds by the positive examples but does not hold by the negative ones. Again, the best specialization is determined by the interestingness measure on the respective confusion matrix.

The search algorithm itself is relatively simple and based on a beam-search. It begins with a pattern which matches all instances. In every iteration the k best-evaluated patterns are further refined by the previously introduced operators. At the end of the iteration, only the k best specializations are preserved for the next loop. The search ends when no more improvements during the last p iterations were made. We do not stop immediately if the present iteration did improve the best rule ($p = 1$) to avoid getting trapped in local minima due to the greedy nature of the approach. The best k templates are reported to the user.

The number of possible patterns grows quickly with the number of alignment lines and predicates, but the number of actually explored patterns is limited by the size of the beam. For each main iteration, the patterns in the beam are extended by the three presented operators. The necessary statistics to find the best specialization can be constructed in $O(n \cdot m^2)$ time where n is the number of cases and m is the number of distinct durations considered as a temporal constraint. Rather than considering every possible duration constraint, the observed durations may be discretized beforehand, thereby limiting the number of choices and the overall runtime.

4 Experimental Evaluation

Artificial example. In order to evaluate the new representation we generated a synthetic dataset on the basis of a pizza recipe. In general the process of making a pizza consists of the following four steps: First we have to mix the ingredients to make the dough. Then we let the dough rise for 60 to 120 minutes in a place without (air) draught. Afterwards we role out the dough and add the toppings. Finally we have to bake the pizza 25 to 29 minutes.

So the dataset consists of the labels: 'make dough', 'let the dough rise', 'role out & coat the dough', 'draught' and 'baking'. Generally each instance is generated as followed: First a 'make dough' interval (5-20 minutes), followed by 'let the dough rise' (60-120 minutes), after 5-30 minutes the 'role out & coat the dough' interval is present (5-10 minutes) and finally after 0-10 minutes there is a 'baking' interval (25-30 minutes). Furthermore it consists of the three classes: perfect pizza (the baking process fits the recipe), pizza burned (the baking process fits the recipe except the baking time is greater than 30 minutes) and dough not rised (the baking process fits the recipe except 'dough rise' is missing or during the time the dough rises there is a draught interval). The interval lengths were chosen randomly and depending on their fit into the abovementioned intervals the class label was selected appropriately.

The training set consists of 99 instances (33 per class) and the test set of 990 instances (330 per class). To compare the results of the new approach with the results of the old approaches we applied the algorithm described in Sect. 3 as well as a reduced version without time-, absent-, exists- and disappears-constraints to simulate the setting of previous approaches.

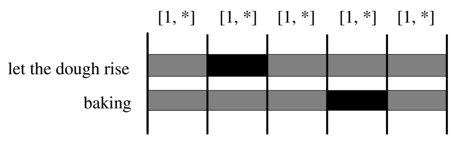


Fig. 8. Pattern for perfect pizza found by the limited pattern language (comparable to earlier approaches)

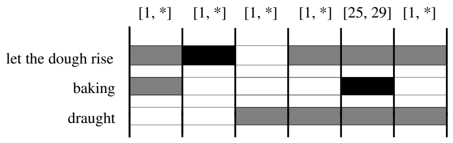


Fig. 9. Pattern for perfect pizza found by new approach.

The best pattern found by the old approach for perfect pizza is shown in Fig. 8. As we can see it requires 'let the dough rise' followed by 'baking' but the pattern neither carries any information on how long the pizza should be backed to get not burned, nor is there a restriction of absent 'draught' during the proving process. The new approach found the pattern shown in Fig. 9, which requires that during the whole time the dough is rising (first two blocks, ends in third block due to absence constraint) there is no draught. Furthermore this pattern demands a baking time between 25 and 29 minutes due to the block 4 to 6, where blocks 4 and 6 forbid the presence of baking and the fifth block requires baking with a duration of 25-29 minutes. Similar patterns were derived for the other classes, too. Figure 10 summarizes how the patterns evaluated on the test set. As we can see the new approach performed considerably better than the old approach, mainly because the old approach lacks means to identify those cases where certain steps of the process were intentionally or incidentally absent.

$$\text{old approach:}\begin{pmatrix} & P & \neg P \\ match & 330 & 490 \\ \neg match & 0 & 170 \end{pmatrix}\begin{pmatrix} & DnR & \neg DnR \\ match & 160 & 0 \\ \neg match & 170 & 660 \end{pmatrix}\begin{pmatrix} & B & \neg B \\ match & 12 & 170 \\ \neg match & 318 & 490 \end{pmatrix}$$

$$\text{new approach:}\begin{pmatrix} & P & \neg P \\ match & 330 & 0 \\ \neg match & 0 & 660 \end{pmatrix}\begin{pmatrix} & DnR & \neg DnR \\ match & 0 & 628 \\ \neg match & 330 & 32 \end{pmatrix}\begin{pmatrix} & B & \neg B \\ match & 330 & 0 \\ \neg match & 0 & 660 \end{pmatrix}$$

Fig. 10. Confusion matrices for the best patterns found by old and new approach. P: perfect pizza, DnR: dough not rised, B: burned pizza

Real data. Furthermore we applied our approach to weather data collected by a weather station located on a small island in the northern sea (Helgoland). This station collected air-pressure, wind strength and wind direction hourly for several years. At first we had to preprocess these timeseries to a stream of intervals, where we used the following labels:

- air-pressure: very low (−−), low (−), middle (o), high (+), very high (++).
- air-pressure slope and wind strength change: highly decreasing (−−), decreasing (−), normal (o), increasing (+), highly increasing (++).

– wind strength: very low (+), low (++), high (+++), very high (++++)
– wind direction: N, NE, E, SE, S, SW, W, NW.

We tried to predict the occurrence of a strong wind (wind strength: high) and extracted all intervals within 72 hours before strong winds occured as positive examples. Negative examples (no strong winds to come) were extracted randomly from other time points. We do not expect to find new knowledge from the data, as it is already known that the actual value of the air pressure is irrelevant, but the change in the air pressure is a good indicator for strong winds, but we are again interested in finding evidence that the *new representation* is useful with real world scenarios.

An obvious pattern that simply requires quite strong winds followed by an increasing trend in wind strength leads to a rule with a J-Measure of 0.0583. Taking the newly introduced constraint types into account, this basic rule can be extended to increase the J-measure by more than 30%. Two of the patterns are shown in Fig. 11. We see that the ability to use the newly introduced constraints allows significant improvements also in real datasets. A much higher J-value is hard to obtain, because the J-measure is limited by the (relatively low) frequency of strong winds.

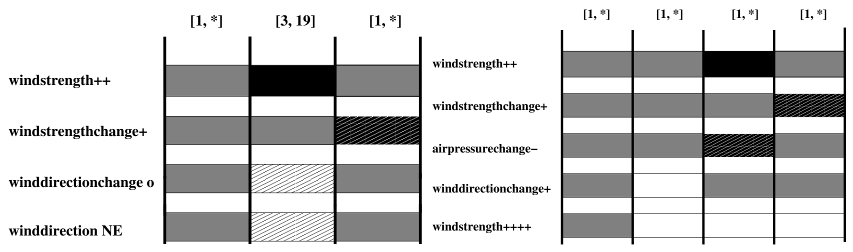


Fig. 11. Two different patterns found to predict a upcoming strong wind.

Furthermore we applied our algorithm to the libras movement data set from the UCI repository [2]. It contains 15 different signs described by their characteristic hand movement over 45 frames, where the current x- and y-positions of the hand were recorded. There are 24 instances per sign, 360 in total. In the first step we have preprocessed the data in order to extract predicates that represent the hand movement. The extracted features address the speed of the hand movement in the x- and y-direction. Although one can easily think of more sophisticated features (rotation, absolute position, ...) and threshold extraction, we used a priori defined thresholds (quantiles) and the following labels only:

– X-Movement: fast left (−−), left (−), constant (o), right (+), fast right (++).
– Y-Movement: fast down (−−), down (−), constant (o), up (+), fast up (++).

For example, a fast hand movement to the upper left is recognized if predicates x-movement−− and y-movement++ hold at the same time.

For every sign we applied the algorithm with and without the newly introduced constraints. Figure 12 shows the the confusion matrices for the best pattern found for each of the two approaches.

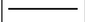
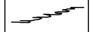
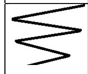
sign	id	old approach	new approach
	1	$\begin{pmatrix} & 1 & \neg 1 \\ match & 22 & 3 \\ \neg match & 2 & 333 \end{pmatrix}$	$\begin{pmatrix} & 1 & \neg 1 \\ match & 23 & 0 \\ \neg match & 1 & 336 \end{pmatrix}$
	7	$\begin{pmatrix} & 7 & \neg 7 \\ match & 12 & 0 \\ \neg match & 12 & 336 \end{pmatrix}$	$\begin{pmatrix} & 7 & \neg 7 \\ match & 22 & 0 \\ \neg match & 2 & 336 \end{pmatrix}$
	9	$\begin{pmatrix} & 9 & \neg 9 \\ match & 0 & 256 \\ \neg match & 24 & 80 \end{pmatrix}$	$\begin{pmatrix} & P & \neg P \\ match & 21 & 0 \\ \neg match & 3 & 336 \end{pmatrix}$
	11	$\begin{pmatrix} & 11 & \neg 11 \\ match & 16 & 6 \\ \neg match & 8 & 330 \end{pmatrix}$	$\begin{pmatrix} & 11 & \neg 11 \\ match & 19 & 0 \\ \neg match & 5 & 360 \end{pmatrix}$
	12	$\begin{pmatrix} & 12 & \neg 12 \\ match & 12 & 0 \\ \neg match & 12 & 336 \end{pmatrix}$	$\begin{pmatrix} & 12 & \neg 12 \\ match & 23 & 0 \\ \neg match & 1 & 336 \end{pmatrix}$
	15	$\begin{pmatrix} & 15 & \neg 15 \\ match & 5 & 270 \\ \neg match & 19 & 66 \end{pmatrix}$	$\begin{pmatrix} & 15 & \neg 15 \\ match & 15 & 0 \\ \neg match & 9 & 336 \end{pmatrix}$

Fig. 12. Effect of newly introduced constraints on libras data [2].

As we can see the patterns with the newly introduced constraints perform considerable better, the number of false positives and false negatives have decreased. On average we have an increase of 3.6 true positives per pattern which equals an increase of 15 percent. For the sign four, which represents a half circle swing (see Fig. 14), we show the identified patterns in Fig. 13. The first pattern was found by the old approach and tries to separate the instances by requiring certain movements – that may occur in other signs also. The second pattern, however, requires a movement to the left over a relatively long period of time (14-24 frames), during which no fast downward movement takes place. Afterwards, an upward movement is required, but the pattern contains no further information/constraints about the last frames, as they are not that helpful for discriminating the signs. The key to the increased performance is the possibility of excluding certain co-occurring predicates.

5 Conclusions

We have proposed a new representation to define templates of evolving variables for classification tasks that overcome deficiencies of previous approaches.

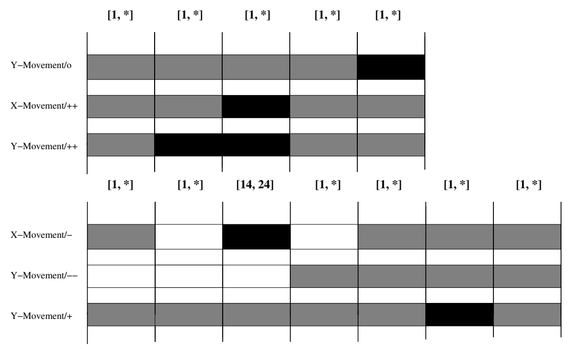


Fig. 13. The two best patterns found by the different approaches to describe sign four (top: old, bottom: new approach).

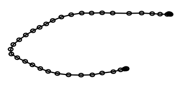


Fig. 14. Instance of sign #4, plotted by connecting all hand positions

A general-to-specific approach to find useful templates has been presented. First experiments were promising and indicated the practical usefulness of the representation. The identified templates may also be used for feature generation when other classifiers shall be applied. Future work includes the application of the approach to business workflows.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
2. A. Asuncion and D. Newman. UCI machine learning repository http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.
3. F. Höppner and F. Klawonn. Finding informative rules in interval sequences. *Intelligent Data Analysis – An International Journal*, 6(3):237–256, 2002.
4. F. Höppner and A. Topp. Classification based on the trace of variables over time. In *Proc. Int. Conf. Intelligent Data Engineering and Automated Learning (IDEAL)*, number 4881 in LNCS, pages 739–749, Birmingham, England, 2007. Springer.
5. P.-S. Kam and A. W.-C. Fu. Discovering temporal patterns for interval-based events. In *Proc. of the 2nd Int. Conf. on Data Warehousing and Knowl. Discovery*, volume 1874 of *LNCS*, pages 317–326. Springer, 2000.
6. F. Mörchen. *Time Series Knowledge Mining.* PhD thesis, Philipps University Marburg, 2006.
7. Y. Shahar and M. A. Musen. RÉSUMÉ: A temporal abstraction system for patient monitoring. *Computers and Biomedical Research*, 26:155–273, 1993.
8. P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Trans. Knowledge Discovery and Engineering*, 4(4):301–316, Aug. 1992.