Learning Temporal Rules from State Sequences[*]

Frank Höppner[**]
Department of Electrical Engineering and Computer Science
University of Applied Sciences, Emden
Constantiaplatz 4
D-26723 Emden, Germany

Abstract

In this paper we consider the problem of learning rules about temporal relationships between labeled time intervals. We learn these rules from a single series of such labeled intervals, which might be obtained from (multivariate) time series by extracting various features of interest, for instance segments of increasing and decreasing local trends. We seek for the identification of frequent local patterns in this *state series*. A temporal pattern is defined as a set of states together with their interval relationships described in terms of Allen's interval logic, for instance "A before B, A overlaps C, C overlaps B" or equivalently "state A ends before state B starts, the gap is covered by state C". In the spirit of association rule mining we propose an algorithm to discover frequent temporal patterns and to generate temporal rules. As an application we consider the problem of deriving local weather forecasting rules that allow us to conclude from the qualitative behaviour of the air-pressure curve to the wind-strength.

1 Introduction

In computer-aided monitoring and control many variables are measured. The plain record of the quantitative values over time does not invoke appreciable levels of cognitive activity to a human. But by simple visual inspection of displayed trends the human operator is capable of controlling the process [Bakshi and Stephanopoulos, 1995]. Skippers use rules that consider the qualitative behaviour of the airpressure curve for short-term local weather forecast [Karnetzki, 1999]. Other examples of rules using qualitative descriptions of time-varying data can be found in the domain of medical diagnosis [Guimarães and Ultsch, 1999], material science [Capelo *et al.*, 1998], or qualitative reasoning [Kuipers, 1994], to mention only a few. Such rules can be derived deductively by means of a good understanding of the underlying process – or inductively by observing the variables for a long period of time. If we lack a good model but have a reasonable amount of data, we might want to go the inductive way. This is a typical knowledge discovery application.

Why qualitative descriptions at all? The problem of finding common characteristics of multiple time series or different parts of the same series requires a notion of similarity. If a process is subject to variation in time (translation or dilation), those measures used traditionally for estimating similarity (e.g. pointwise Euclidean norm) will fail in providing useful hints about the time series similarity in terms of the cognitive perception of a human. It seems that a human breaks a time series into suitable segments, such that all points in each segment behave similar or follow the same local trend. Each of these segments is usually simple in shape and easy to grasp. The human labels or classifies them into a small number of primitive shapes or patterns. Matching of time series is then performed on the basis of these labeled segments rather than on the raw time series. The primitive patterns can be defined a priori (for example "slightly increasing segment") [Bakshi and Stephanopoulos, 1995; McIlraith, 1989; Capelo *et al.*, 1998], can be learned from a set of examples (labeled training set) [Guimarães and Ultsch, 1999], or can be found automatically by means of clustering short subsequences [Das *et al.*, 1998]. Finally, we arrive at a sequence of labeled intervals: time intervals in which a certain condition holds in the original time series.

This paper considers the problem of discovering temporal relationships between primitive patterns in time series in a fairly general manner: A temporal pattern consists of a number of states (the primitive patterns) and their temporal relationship in terms of Allen's temporal logic [Allen, 1983]. In the sequence of labeled intervals, we seek for frequent patterns in a fashion that is similar to the discovery of association rules [Agrawal *et al.*, 1996], which has been extended to event sequences in [Mannila *et al.*, 1997]. Given the frequent patterns, rules about temporal relationships can be derived. As an application of this algorithm, we consider the problem of finding rules about the qualitative behaviour in multivariate time series.

The outline of the paper is as follows: In section 2 we define our notion of a state sequence. A subset of state intervals in a state sequence can be characterized by means of their relative positions to each other. This leads us to the definition of a temporal pattern in section 3. Next, we consider the question

[*]This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. Kl 648/1.

[**]Contact: frank.hoeppner@ieee.org

how often a pattern occurs in the state sequence in section 4. Adapting ideas from the discovery of association rules we propose an algorithm to discover temporal patterns in section 5. An application example is given in section 6, before we come to the conclusions in section 7.

2 State Sequences

Let $\mathcal{S}$ denote the set of all possible trends, properties, or states that we want to distinguish, for example "pressure goes down" or "water level is constant". A state $s \in \mathcal{S}$ holds during a period of time $[b, f)$ where b and f denote the *initial point* in time when we enter the state and the *final point* in time when the state no longer holds. A state sequence on $\mathcal{S}$ is a series of triples defining state intervals

$$(b_1, s_1, f_1), (b_2, s_2, f_2), (b_3, s_3, f_3), (b_4, s_4, f_4), \ldots$$

where $b_i \leq b_{i+1}$ and $b_i < f_i$ holds. We do not require that one state interval has ended before another state interval starts. This enables us to mix up several state sequences (possibly obtained from different sources) into a single state sequences.
However, we do require that every state (b_i, s, f_i) is *maximal* in the sense, that there is no (b_j, s, f_j) in the series such that $[b_i, f_i)$ and $[b_j, f_j)$ overlap or meet each other:

$$\forall (b_i, s_i, f_i), (b_j, s_j, f_j), i < j : f_i \leq b_j \Rightarrow s_i \neq s_j \qquad (1)$$

If (1) is violated, we can merge both state intervals and replace them by their union $(\min(b_i, b_j), s, \max(f_i, f_j))$.
As an example, we could have classified the points in a time series into qualitative states "increasing", "decreasing", and "constant". These three states partition the time series completely, that is, any state is continued without gap by another state. But it is also possible to use only primitive patterns like "increasing" and "highly increasing". The missing patterns (i.e. "decreasing") will cause some gaps in the description of a particular time series, but this does not hinder the analysis of the state sequence. In what follows, we assume that such a state sequence is given.

3 Temporal Patterns

We use Allen's temporal interval logic [Allen, 1983] to describe the relation between state intervals. For any pair of intervals we have 13 possible relationships; they are illustrated in Figure 1. For example, we say "A meets B" if interval A terminates at the same point in time at which B starts. The inverse relationship is "B is-met-by A". In the following we will abbreviate the set of interval relations as shown in the figure by $\mathcal{I}$.
Given n state intervals (b_i, s_i, f_i), $1 \leq i \leq n$, we can capture their relative positions to each other by an $n \times n$ matrix R whose elements $R[i, j]$ describe the relationship between state interval i and j. As an example, let us consider the state sequence in Figure 2. Obviously state A is always followed by B. And the lag between A and B is covered by state C. Below the state interval sequence both of these patterns are written as a matrix of interval relations. Formally, a temporal pattern of size n is defined by a pair (s, R), where $s : \{1, .., n\} \to \mathcal{S}$ maps index i to the corresponding state,

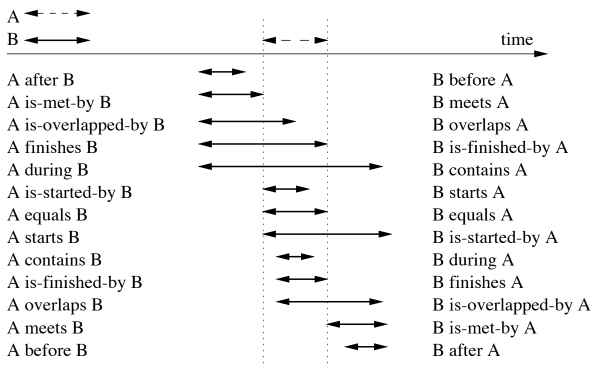
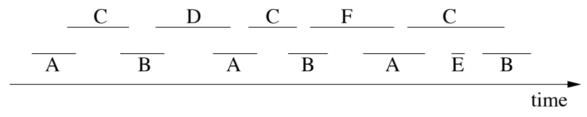


Figure 1: Allen's interval relationships.

and $R \in \mathcal{I}^{n \times n}$ denotes the relationship between $[b_i, f_i)$ and $[b_j, f_j)$[1]. By $\dim(P)$ we denote the dimension (number n of intervals) of the pattern P. If $\dim(P) = k$, we say that P is a k-pattern. Of course, many sets of state intervals map to the same temporal pattern. We say that the set of intervals $\{(b_i, s_i, f_i) \,|\, 1 \leq i \leq n\}$ is an *instance* of its temporal pattern (s, R). We define the space $TP(\mathcal{S})$ of temporal patterns over $\mathcal{S}$ informally as the space of all valid temporal patterns of arbitrary dimension[2].

state interval sequence:



temporal relations:

	A	B
A	=	b
B	a	=

	A	B	C
A	=	b	o
B	a	=	io
C	io	o	=

(abbreviations: a=after, b=before, o=overlaps, io=is-overlapped-by)

Figure 2: Example for state interval patterns expressed as temporal relationships.

3.1 Partial Order on Temporal Patterns

Next, we define a partial order $\sqsubseteq$ on temporal relations. We say that temporal relation (s_A, R_A) is subpattern of (s_B, R_B) (or $(s_A, R_A) \sqsubseteq (s_B, R_B)$), if $\dim(s_A, R_A) \leq \dim(s_B, R_B)$ and there is an injective mapping $\pi : \{1, .., \dim(s_A, R_A)\} \to \{1, .., \dim(s_B, R_B)\}$ such that

$$\forall i, j \in \{1, .., \dim(s_A, R_A)\} : R_A[i, j] = R_B[\pi(i), \pi(j)]$$

The relation $\sqsubseteq$ is reflexive and transitive, but not antisymmetric: we can have $(s_A, R_A) \sqsubseteq (s_B, R_B)$ and $(s_B, R_B) \sqsubseteq (s_A, R_A)$ without $s_A = s_B$ and $R_A = R_B$ due to a different state ordering. But permutating the states does

[1]To determine the interval relationships we assume closed intervals $[b_i, f_i]$

[2]Conditions for a *valid* temporal pattern are, for instance, that $R[i, j]$ is always the inverse of $R[j, i]$.

not change the semantics of the temporal pattern. Therefore, we define $(s_A, R_A) \equiv (s_B, R_B) :\Leftrightarrow (s_A, R_A) \sqsubseteq (s_B, R_B) \wedge (s_B, R_B) \sqsubseteq (s_A, R_A)$ and consider the factorisation $({}^{TP(\mathcal{S})}/_{\equiv}, {}^{\sqsubseteq}/_{\equiv})$, where $\sqsubseteq$ has been generalized canonically to equivalence classes. Then, ${}^{\sqsubseteq}/_{\equiv}$ is also antisymmetric and thus a partial order on (equivalence classes of) temporal patterns.

3.2 Normalized Form of a Temporal Pattern

To simplify notation we pick a subset $NTP(\mathcal{S}) \subseteq TP(\mathcal{S})$ of *normalized* temporal patterns such that $NTP(\mathcal{S})$ contains one element for each equivalence class of ${}^{TP(\mathcal{S})}/_{\equiv}$ and $(NTP(\mathcal{S}), \sqsubseteq)$ is isomorphic to $({}^{TP(\mathcal{S})}/_{\equiv}, {}^{\sqsubseteq}/_{\equiv})$. In the remainder, we will then use $(NTP(\mathcal{S}), \sqsubseteq)$ synonymous to $({}^{TP(\mathcal{S})}/_{\equiv}, {}^{\sqsubseteq}/_{\equiv})$.

The intuitive idea is to order the state intervals in time with increasing index. However, this ordering is slightly more complex with arbitrary intervals than with points. Given an ordering of states, we say that a temporal pattern (s_A, R_A) *has normalized form*, if for all $1 \leq i \leq \dim(s_A, R_A)$ and $i < j \leq \dim(s_A, R_A)$ the following conditions hold:

- *Case 1:* $s(i) = s(j)$. If we have two intervals with identical states, then by the maximality assumption (1) there must be a time gap between the intervals, otherwise we could merge both state intervals into a single new one, which contains both intervals. Therefore, in this case we have only two possible relations $R[i,j] \in \{$*before*, *after*$\}$. To preserve temporal ordering we require $R[i,j] =$*before*.
- *Case 2:* $s(i) \neq s(j)$.
  - *Case 2a: distinct initial times.* If the initial times of both intervals are different, we use the ordering of the initial times, that is, in a normalized form we have $R[i,j] \in \{$*contains*, *is-finished-by*, *overlaps*, *meets*, *before*$\}$ (cf. Figure 1).
  - *Case 2b: initial times coincidence.* Thus we have $R[i,j] \in \{$*equals*, *starts*, *is-started-by*$\}$. If both intervals are identical, we use the order on the states, that is, in a normalized form we require $s(i) < s(j)$ (note that we are sure that $s(i) \neq s(j)$ in this subcase). If the final times are different, we require $R[i,j] =$ *is-started-by* to make sure that the interval with index i ends before the interval with index j.

4 Occurrences of Temporal Patterns in State Sequences

To be considered interesting, a temporal pattern is limited in its extension, that is, the whole pattern has to be small enough to be observed by an operator. We therefore choose a maximum duration t_{max}, which serves as the width of a sliding window which is moved along the state sequences. We consider only those pattern instances that can be observed within this window. In a monitoring and control application, this threshold could be taken from the maximum history length that can be displayed on the monitor and thus be inspected by the operator.

Note, that this does not necessarily mean, that the end points of a set of state intervals $P := \{(b_i, s_i, f_i) \mid 1 \leq i \leq n\}$, differ by no more than t_{max}. Denoting the temporal extent (or duration) of P by

$$D(P) := \max\{f \mid (b,s,f) \in P\} - \min\{b \mid (b,s,f) \in P\},$$

we do not have $D(P) \leq t_{max}$ in general. Figure 3 illustrates this fact in case there are intervals with a length that exceeds t_{max} (window drawn with solid lines). State A lasts for a time period that is longer than t_{max}, nevertheless we can observe the pattern "D after C, A contains C and D" within the window. The pattern "B before C" in the window drawn with dashed lines is another example where we can observe the pattern although $D(P) > t_{max}$. However, we can not (yet) observe "A contains C" in the dashed window, because the final time of C is not yet visible – the pattern may also represent an *overlaps* or *finishes* relation.

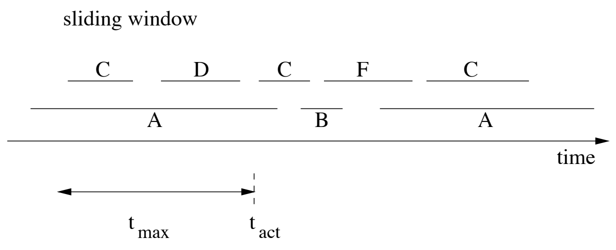


Figure 3: Sliding a window of width t_{max} along the state sequence.

We define the total time in which the pattern can be observed within the sliding window as the support $\text{supp}(P)$ of the pattern P. Let us illustrate this definition with some examples in Figure 4. In subfigure (a) we have a single state B. We see the pattern for the first time, when the right bound of the sliding window touches the initial time of the state interval (dotted position of sliding window). We can observe B unless the sliding window reaches the position that is drawn with dashed lines. The total observation time is therefore the length of the sliding window t_{max} plus the length of state interval B. The support (observation duration) is depicted at the bottom of the subfigure.

Subfigure (b) shows another example "A overlaps B". We can observe an instance of the pattern as soon as we can see state B and we loose it when A leaves the sliding window. If the pattern occurs multiple times, two things may happen: If there is a gap between the pattern instances, such that we loose the pattern in the meanwhile, then the support of the individual instances add up to the support of the pattern, as shown in subfigure (c). If there is no such gap (subfigure (d)), we see the pattern as soon as a first instance enters the sliding window until the last instance leaves the window. In the meantime, it does not matter *how many* instances are present, as long as there is at least one.

If we divide the support of a pattern by the length of the state sequence plus the window width t_{max} we obtain the relative frequency p of the pattern: If we randomly select a window

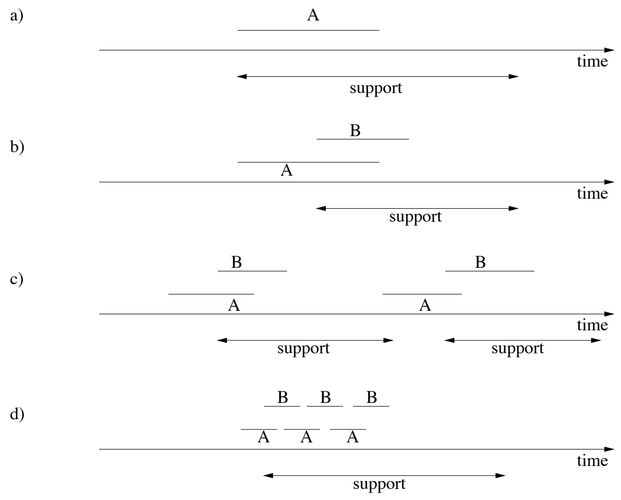


Figure 4: Illustration of our notion of support.

position we can observe the pattern with probability p. Also note that there is no need for discretization, we can handle time continuously by jumping from interval bound (initial or final time) to interval bound and integrating the support over the jump period. This is because observability of a pattern changes only if the sliding window meets one of the interval bounds.

5 Discovery of temporal rules

A pattern is called *frequent*, if its support exceeds a threshold supp$_{min}$. The task is to find all frequent temporal patterns in $NTP(\mathcal{S})$, from which we then create the temporal rules. To find all frequent patterns we start in a first database pass with the estimation of the support of every single state (also called candidate 1-patterns). After the kth run, we remove all candidates that have missed the minimum support and create out of the remaining frequent k-patterns a set of candidate $(k+1)$-patterns whose support will be estimated in the next pass. This procedure is repeated until no more frequent patterns can be found. The fact that the support of a pattern is always greater or equal to the support of any of its subpatterns

$$\forall \text{patterns } P, Q: \quad Q \sqsubseteq P \ \Rightarrow\ \text{supp}(Q) \geq \text{supp}(P) \qquad (2)$$

guarantees that we do not miss any frequent patterns. At this level of detail the procedure is identical to association rule mining [Agrawal *et al.*, 1996].

5.1 Candidate Generation

The number of potential candidates grows exponentially with the size k of the patterns. Efficient pruning techniques are therefore necessary to keep the increase in the number of candidates moderate. We use three different pruning techniques. The technique that is used for the discovery of association rules [Agrawal *et al.*, 1996] can still be applied to temporal patterns: Due to (2), every k-subpattern of a $(k+1)$-candidate must be frequent, otherwise the candidate itself cannot be frequent. To enumerate as few non-candidate $(k+1)$-patterns

R_P	$s_P(0)\ldots s_P(k-1)$	p
$s_P(0)$ ⋮ $s_P(k-1)$	A	B
p	C	$=$

(a) Pattern P

R_Q	$s_Q(0)\ldots s_Q(k-1)$	q
$s_Q(0)$ ⋮ $s_Q(k-1)$	A	D
q	E	$=$

(b) Pattern Q

R_X	$s_X(0)\ldots s_X(k-1)$	p	q
$s_X(0)$ ⋮ $s_X(k-1)$	A	B	D
p	C	$=$	r
q	E	ir	$=$

(c) New pattern X

Figure 5: Generating a candidate $(k+1)$-pattern X out of two k-patterns P and Q that are identical when restricted to the first $k-1$ states.

as possible, we join any two frequent k-patterns P and Q that share a common $(k-1)$-pattern as a prefix. Let us denote the remaining states in P and Q besides those in the prefix as p and q respectively. We denote the interval relationship between p and q in the candidate pattern $X = (s_X, R_X)$ as $R_X[k, k+1] = r$. Figure 5 illustrates how to build the $(k+1)$-pattern matrix R_X out of R_P and R_Q. Since R_P and R_Q are identical with respect to the first $k-1$ states in normalized form, the same is true for the new pattern X (indicated by the same submatrix A). The relationship between p and q and the first $k-1$ states can also be taken from R_P and R_Q. Thus, as we can see in Figure 5(c), the only degree of freedom is r. From the $(k-1)$-pattern prefix and the two states p and q we thus can build up a $(k+1)$-pattern which is completely specified up to the relation between p and q.

The freedom in choosing r yields 13 different patterns that might become candidate $(k+1)$-patterns, because there are 13 possible interval relationships. Since we can restrict ourselves without loss of generality to normalized patterns X, the number of possible values for r reduces to a maximal number of 7. Before we check each of the seven $(k+1)$-patterns for frequent k-subpatterns, we apply another pruning technique based on the law of transitivity. For example, the two 2-patterns “A meets B” and “A meets C” share the primitive 1-pattern “A” as a common prefix. We have to fix the missing relationship between B and C to obtain a 3-candidate. The law of transitivity for interval relations [Allen, 1983] tells us that the possible set of interval relations is $\{$*is-started-by*, *equals*, *starts*$\}$. In normalized form, only 2 out of 7 possible relationships remain. In general, for each state $s(i)$ of the first $k-1$ states we apply Allen's transitivity table to the relationship between p and $s(i)$ ($R_P[k,i]$) and $s(i)$ and q ($R_Q[i,k]$). Only those values for r that do not contradict the results of the $k-1$ applications of the transitivity table yield a candidate pattern.

Finally, for every temporal pattern Q we maintain an *observed* and *expected support set* O_Q and E_Q, resp. The set O_Q contains all points in time that contribute to the support of the pattern Q, that is, all points in time in which the pattern can be observed in the sliding window. Before we consider a $(k+1)$-pattern P as a candidate pattern, we intersect[3] all sets O_Q of all k-subpatterns Q of P. The result gives us the expected support of P in E_P. The cardinality of E_P serves as a tighter upper bound of the support of P than $\min\{E_Q \mid Q \sqsubseteq P, \dim(Q)=k\}$ does. If it stays below supp_{min} the pattern cannot become a frequent pattern, therefore we do not consider it as a candidate.

5.2 Support Estimation

In order to estimate the support for the candidate patterns, we sweep through the state sequence and incrementally update the list of states which are currently visible in the sliding window. We also update the relation matrix for the states in the sliding window incrementally. By t_{act} we denote the right bound of the sliding window.

The set of candidate patterns is partitioned into three subsets, which we call the set of passive, active, and potential candidates. The set of passive candidates contains those candidates P that we do not expect in the current sliding window because the expected support does not contain the time of the current window position, that is, $t_{act} \notin E_P$. The set of potential candidates contains those candidates for which we have $t_{act} \in E_P$, that is, there is a chance of observing P in the window. Finally, the set of active patterns contains those patterns that are currently observable in the sliding window.

At the beginning all patterns are passive patterns. Associated with every pattern we have the set of expected support E_P, we therefore know in advance when the pattern will become a potential pattern, namely at *activation time* $a_P = \min\{t \mid t \in E_P\}$. If the set E_P is organized as a sorted list of intervals, the minimum is simply the left bound of the first interval in the list. We keep the set of passive patterns ordered by their activation time. Whenever t_{act} reaches the activation time of a pattern P, P becomes either a potential or active pattern, depending on whether P occurs in the sliding window or not. When P becomes a potential pattern, we remove the leading interval from the E_P list and store the *deactivation time* d_P (end of the interval), because at that time the pattern will fall back into the set of passive patterns.

A potential pattern P becomes a passive pattern if the fall back-time d_P has been reached by the sliding window. Whenever a new state interval enters the sliding window, we check for all potential patterns if an instance of the pattern can be found. If this is the case, the potential pattern becomes an active pattern, otherwise we keep it as a potential pattern. If a pattern instance has been found, we calculate the point in time when the pattern disappears and use it as the fall back-time for the active pattern.

Just like the set of passive patterns, the set of active patterns is sorted by their fall back-times. Whenever t_{act} reaches the fall back-time of an active pattern, we check whether a new pattern instance has entered the sliding window in the meanwhile. In this case the pattern remains an active pattern, but we update the fall back-time. Otherwise, depending on whether $d_P < t_{act}$ or not, the active pattern becomes a potential or passive pattern.

Whenever a pattern instance has been found, the support of the pattern is incrementally updated, that is, we insert the period of pattern observation (the support) into O_S. Since we have an upper bound of the remaining support (namely the cardinality of the continuously updated set E_P), we can perform a fourth online pruning test. If the support achieved so far ($\text{card}(O_P)$) plus the maximally remaining support ($\text{card}(E_P)$) drops below supp_{min} we do not consider the pattern any longer. At the end of each database pass, the set E_P is empty and O_P contains the support of P, which is then subsequently used in the next candidate generation step for pruning.

5.3 Rule Generation

After having determined all frequent temporal patterns, we can construct rules $X \mapsto Y$ from every pair (X, Y) of frequent temporal patterns with $X \sqsubseteq Y$. We restrict ourselves to "forward rules", that is, rules that make conclusions in the future rather than in the past. If the confidence of the rule $\text{conf}(A \rightarrow B) = \frac{\text{supp}(B)}{\text{supp}(A)}$ is greater than the minimal confidence, the rule is printed. Enumeration of all possible rules can be done efficiently using techniques described in [Agrawal *et al.*, 1996].

5.4 Disjunctive Combination of Temporal Patterns

When analysing the rules obtained by the algorithm, we must keep in mind that we were seeking for the simple interval relationships only, that is, those relationships that consist of a single attribute $r \in \mathcal{I}$. If a process B is started some time after A has started, then this can result in a number of rules "$A \rightarrow B$" with temporal relationsships *overlaps*, *meets*, and *before*. The confidence of the *true* relationship (which is in this case: A *overlaps/meets/before* B) might be very high, but the confidence values we observe for the three rules we have found are comparatively low. We are not allowed to add up the confidence values of all three rules in order to obtain the confidence of the composed rule. This would lead to an overestimation, because there might be sliding windows that contain multiple of these patterns simultaneously, and in this case we would count them twice (or more).

Theoretically, we could also consider the more complex relationships during the discovery process, but probably the combinatorial explosion of candidate patterns cannot be pruned appropriately any longer (2^7 possible interval relationships in

[3]The sets O_Q and E_Q can be organized as lists of intervals. The intersection is also a list of intervals. We only have to add up the interval lengths to obtain the cardinality.

normalized form instead of 7). Fortunately, it is not necessary to consider all these combinations during the execution of the algorithm, since we can calculate the support of composed rules afterwards. The support of a pattern P that is the disjunction of two patterns Q and R can be calculated easily as $\text{supp}(P) = \text{card}(O_Q \cup O_R)$. The sets of observed support O_Q and O_R have been calculated already during the execution of the algorithm, all we have to do is to store the sets for later access.
(Note that we cannot guarantee that we will find all frequent pattern compositions in this way. Several patterns that do not reach supp_{min} individually might fulfill this requirement after their combination.)

6 Evaluation and Discussion

We have examined air-pressure and wind strength/wind direction data from a small island in the northern sea[4]. From the time stamps we have also extracted the season. It is well known that local differences in air pressure are the cause for wind, therefore we should find some relationships between these variables. The features have been measured hourly and we used three years of data from 1981-1983.
We have applied kernel smoothing in order to compensate for noise and to get more robust estimates of the first and second derivative. Then, the smoothed series have been partitioned into primitive patterns. To encourage meaningful findings of temporal patterns, we tried to simulate the way a human would partition the time series. In a first stage, the air pressure curve has been segmented into increasing, level, and decreasing segments. Among the increasing segments, if the derivative is larger than 50% of all values measured for increasing derivatives, we refine them by an additional state "highly increasing", if it is larger than 80% we speak of "very highly increasing". As an alternative to this *overlapping* state definition, we could have defined an exclusive partition "very highly incr.", "highly incr.", "increasing", etc. However, having chosen the threshold values heuristically, we cannot be sure that we have chosen them meaningful (with respect to some patterns we want to discover). As we will see in the following example, if we are not sure about the threshold values used to define the states, the hierarchical definition is preferable over the exclusive. There is a pattern "highly increasing segment meets level segment meets highly decreasing segment" in the time series depicted in Figure 6. If the threshold values for the derivatives have not been chosen appropriately, the increasing flank of the second wave will not be classified as "highly increasing". But if we use the state series "A" in Figure 6, we will at least discover the pattern "increasing segment meets level segment meets highly decreasing segment". If we choose the exclusive state definition "B" in Figure 6, the depicted state series contributes only partially to the support of both discussed patterns. If we have badly chosen some threshold values, with a hierarchical state definition we can at least be sure that we will find a *similar pattern* in terms of the employed state hierarchy.
In addition to states that characterize the slope, we used some states that address the second derivative of the air pressure curve. High values in the second derivative can be used to distinguish sharp peaks from flat hills, for example. To simplify the notation, we will use the following state abbreviations: `dec`, `lvl`, `inc` for decreasing, constant, and increasing trends, respectively. `ccv` and `cvx` denotes concave and convex curvature. An additional suffix `-high` is used for highly increasing (`inc-high`) or higly concave (`ccv-high`) segments, etc. A suffix `-w`, `-p`, `-d` indicates that the segments refers to the wind strength, air pressure, or wind direction curve, respectively.

[4]Helgoland, 54:11N 07:54O

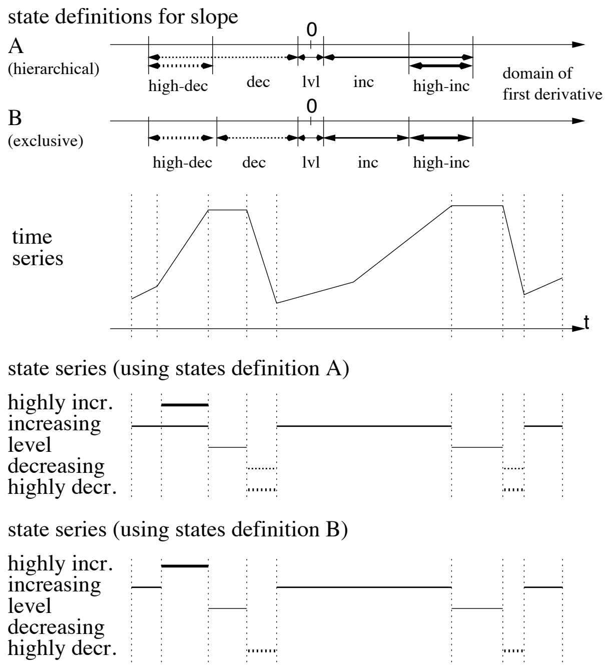
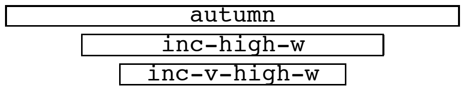


Figure 6: Two ways of partitioning a time series.

The window width t_{max} was chosen to be 48 hours, $\text{supp}_{min} = 2\%$ of the total time of three years, $\text{conf}_{min} = 40\%$. The calculation took 18 minutes on a 64MB laptop computer with a Pentium II Mobile processor running Linux. As usually in the context of rule mining, we have found a large number of rules. Due to lack of space, here are only some exemplary rules.
Autumn is known for its strong winds, and thus we can find rules like this:

$$\texttt{autumn}, \texttt{inc-high-w} \rightarrow \texttt{inc-v-high-w}$$

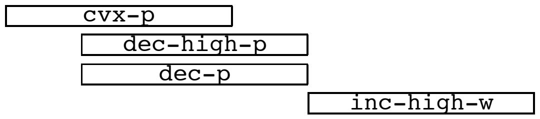
with the temporal relationship `autumn` *contains* `inc-high-w`, and `inc-high-w` *contains* `inc-v-high-w`, which might be depicted as



The confidence of this rules is about 51% and it can be applied in 15% of all sliding window positions. If the minimum confidence is small enough, we obtain the same rule for all other seasons, too, but with much lower confidence.
As expected from [Karnetzki, 1999], we have also found many rules that reflect the relationship between change in air

pressure and change in wind strength. Here is a rule to forecast highly increasing wind strength that can be applied in 5% of the time series with a confidence value of 94%:

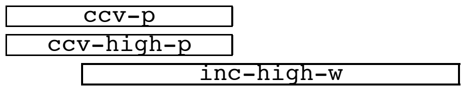
$$\texttt{cvx-p}, \texttt{dec-high-p}, \texttt{dec-p} \rightarrow \texttt{inc-high-w}$$



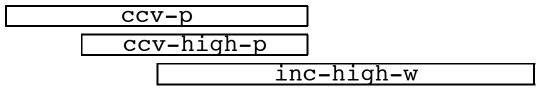
An example for the phenomenon described in Section 5.4 is the following rule:

$$\texttt{ccv-p}, \texttt{ccv-high-p} \rightarrow \texttt{inc-high-w}$$

with the temporal relationships depicted by



and



The two rules differ in the premise, where we have an *equals* relation in the first rule (confidence 55%, applicability 3.7%) and a *is-finished-by* relation in the second rule (confidence 45%, applicability 5.6%). A disjunctive combination of both rules increases the confidence significantly. Interestingly, there is *no* rule "$\texttt{ccv-high} \rightarrow \texttt{inc-high-w}$" (when using a confidence level of 40%). This emphasizes the importance of the concave segment, which in both rules is finished by the highly concave segment. If there is a highly concave segment *during* a concave segment, there seems to be a much lower probability for highly increasing winds in this pattern. Also note that we would not have detected these rules if we had chosen an exclusive state definition.

However, with many rules the confidence values are comparatively low. This is not necessarily because the conclusion of a rule was not present, but may also come from the fact that longer patterns (premise and conclusion) get lower support values than shorter patterns (premise only). Compensating this effect is a major topic for future work.

7 Conclusion

We have proposed a technique for the discovery of temporal rules in state sequences, which might stem from multivariate time series for instance. The examples in section 6 have shown that the proposed method is capable of finding meaningful rules that can be used as rules-of-thumb by a human, but also in a knowledge-based expert system. The rules can be easily interpreted by a domain expert, who can verify the rules by means of its background knowledge or use them as an inspiration for further investigation. Even if there is already considerable background knowledge, the application of this method might be valuable, for example if the known rules incorporate more variables than available in a specific technical system. For example, weather forecasting rules as discussed by Karnetzki [1999] also use information about the general weather outlook (cloudiness) or information from the local weather forecasting station. Such information might be difficult to incorporate or expensive to measure, and in such a case one is interested in how much one can achieve by just using the available variables.

As our next step, we will focus on the enrichment of the qualitative rules with quantitative (discriminating) values, for example: "if the air pressure falls quickly for more than 2 hours, the wind strength will increase within 1 hour."

Acknowledgements I would like to thank Prof. Frank Klawonn for fruitful discussions, and the Deutsche Wetterdienst for providing the data.

References

[Agrawal *et al.*, 1996] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In [Fayyad *et al.*, 1996], chapter 12, pages 307–328. MIT Press, 1996.

[Allen, 1983] James F. Allen. Maintaing knowledge about temporal intervals. *Comm. ACM*, 26(11):832–843, 1983.

[Bakshi and Stephanopoulos, 1995] Bhavik R. Bakshi and George Stephanopoulos. Reasoning in time: Modeling, analysis, and pattern recognition of temporal process trends. In *Advances in Chemical Engineering*, volume 22, pages 485–548. Academic Press, Inc., 1995.

[Capelo *et al.*, 1998] Antonio C. Capelo, Liliana Ironi, and Stefania Tentoni. Automated mathematical modeling from experimental data: An application to material science. *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, 28(3):356–370, August 1998.

[Das *et al.*, 1998] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 16–22. AAAI Press, 1998.

[Fayyad *et al.*, 1996] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[Guimarães and Ultsch, 1999] Gabriela Guimarães and Alfred Ultsch. A method for temporal knowledge conversion. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Advances in Intelligent Data Analysis, Proc. of the 3rd Int. Symp.*, pages 369–380, Amsterdam, The Netherlands, 1999. Springer, Berlin.

[Karnetzki, 1999] Dieter Karnetzki. *Luftdruck und Wetter*. Delius Klasing, 3 edition, 1999.

[Kuipers, 1994] Benjamin Kuipers. *Qualitative Reasoning – Modeling and Simulation with Incomplete Knowledge*. MIT Press, 1994.

[Mannila *et al.*, 1997] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. Technical Report 15, University of Helsinki, Finland, February 1997.

[McIlraith, 1989] Sheila A. McIlraith. Qualitative data modeling: application of a mechanism for interpreting graphical data. *Computational Intelligence*, 5:111–120, 1989.