Discovery of Temporal Patterns[⋆]

Learning Rules about the Qualitative Behaviour of Time Series

Frank Höppner

Department of Electrical Engineering and Computer Science
University of Applied Sciences, Emden
Constantiaplatz 4
D-26723 Emden, Germany
frank.hoeppner@ieee.org

Abstract. Recently, association rule mining has been generalized to the discovery of episodes in event sequences. In this paper, we additionally take durations into account and thus present a generalization to *time intervals*. We discover frequent temporal patterns in a single series of such labeled intervals, which we call a state sequence. 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". As an example 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. Here, the states have been extracted automatically from (multivariate) time series and characterize the trend of the time series locally within the assigned time interval.

1 Introduction

To predict or forecast a system's behaviour in the near future it is probably best to develop a global model of the system and to estimate its parameters with the help of observations in the past. But the identification of such a model requires substantial knowledge about the whole system, which is absent in typical knowledge discovery applications. Nevertheless, we often expect certain relationships between measured variables and the systems behaviour in the future, may be we have already some snapshots of typical behaviour in mind, but we are far away from being able to model the system as a whole. Such *typical key situations* are often associated with a typical qualitative behaviour of measured variables, and consequently humans control technical systems often by simple visual inspection of displayed trends [4]. Examples of rules using qualitative descriptions of time-varying data can be found in the domain of medical diagnosis, material science

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

[6], diagnostics and supervision [11] or qualitative reasoning [12], to mention only a few. In this paper, we consider the problem of deriving such *local* rules inductively by observing the variables for a long period of time.

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. This problem has been addressed by many authors in the literature, e.g. [1, 5]. Here we use qualitative descriptions to divide up the time series in small segments, each of it easy to grasp and understand by the human. Matching of time series is then performed on the basis of these labeled segments rather than on the raw time series. The basic descriptions can be defined a priori (for example "slightly increasing segment") [4, 14, 6], can be learned from a set of examples (labeled training set), or can be found automatically by means of clustering short subsequences [7]. 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: The time series is turned into a sequence of labeled intervals in Sect. 2. A temporal pattern will be defined as a number of states (the primitive patterns) and their temporal relationship in terms of Allen's temporal logic [3] in Sect. 3. After discussing how to count patterns in an interval sequence in Sect. 4, we seek for frequent patterns in Sect. 5 in a fashion that is similar to the discovery of association rules [2], which has been extended to event sequences in [13, 15]. 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 of multivariate time series in Sect. 6.

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:

$$\forall (b_i, s_i, f_i), (b_j, s_j, f_j), i < j : f_i \geq b_j \Rightarrow s_i \neq s_j \tag{1}$$

If (1) is violated, we can merge both state intervals and replace them by their union $(\min(b_i, b_j), s, \max(f_i, f_j))$.

3 Temporal Patterns

We use Allen's temporal interval logic [3] to describe the relation between state intervals. For any pair of intervals we have 13 possible relationships; they are illustrated in Fig. 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 denote the set of interval relations as shown in the figure by $\mathcal{I}$.

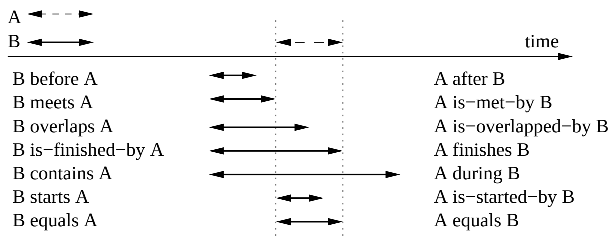
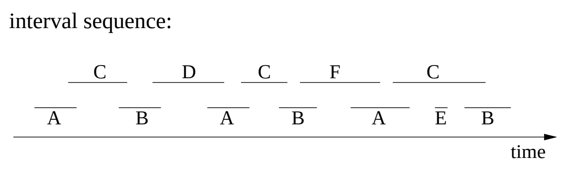


Fig. 1. Allen's interval relationships.

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 Fig. 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 $s : \{1, .., n\} \rightarrow \mathcal{S}$ maps index i to the corresponding state, 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 $\{(b_i, s_i, f_i) \mid 1 \leq i \leq n\}$ 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].

[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 $R[j, i]$.



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)

Fig. 2. Example for state interval patterns expressed as temporal relationships.

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 π : $\{1, .., \dim(s_A, R_A)\} \rightarrow \{1, .., \dim(s_B, R_B)\}$ such that

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

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 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.

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 $(NTP(\mathcal{S}), \sqsubseteq)$ synonymously to $({}^{TP(\mathcal{S})}/_{\equiv}, {}^{\sqsubseteq}/_{\equiv})$. Within each equivalence class, we can order the patterns lexicographically by initial time, final time, and state. This ordering is unique thanks to (1). We use the first pattern in this ordering as the representative of the class.

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 a (forgetful) 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.

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. (Space limitations prohibit the justification of this choice, we refer the interested reader to [9].) Let us illustrate this definition with some examples in Fig. 3. In subfigure (a) we have a single state A. 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 A 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 A. 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.

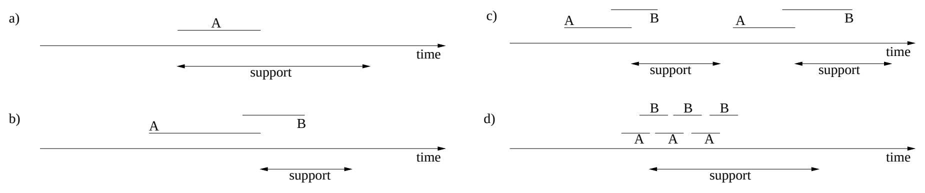


Fig. 3. Illustration of our notion of support.

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 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 less than or equal to the support of any of its subpatterns

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

guarantees that we do not miss any frequent patterns. At this level of detail the procedure is identical to association rule mining [2].

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 [2] 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 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 4 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 Fig. 4(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.

R_P	s_P $0 \dots k-1$	p
s_P $0 \vdots k-1$	A	B
p	C	$=$

(a) Pattern P

R_Q	s_Q $0 \dots k-1$	q
s_Q $0 \vdots k-1$	A	D
q	E	$=$

(b) Pattern Q

R_X	s_X $0 \dots k-1$	p	q
s_X $0 \vdots k-1$	A	B	D
p	C	$=$	r
q	E	ir	$=$

(c) New pattern X

Fig. 4. 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.

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, 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 [3] 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

Again, due to space limitations we can give only a quick overview of the basic ideas, a more detailed report is currently in preparation (contact the author).

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

[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.

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. (Since the set of potential pattern may become quite large, this is the most expensive operation.) 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 updated incrementally, 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 (card(O_P)) plus the maximally remaining support (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 [2].

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 relationships *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). Fortunately, it is possible to calculate the support of composed rules afterwards. The support of a pattern P which is a 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 fulfil 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. Although global weather forecast is (more or less) done perfectly by large-scale weather simulations, it is still not possible to precisely localize where a certain weather phenomenon will occur to which extent at what time. Rules about the qualitative behaviour of the air pressure curve indeed help sailors in short-term local weather forecasting [10].

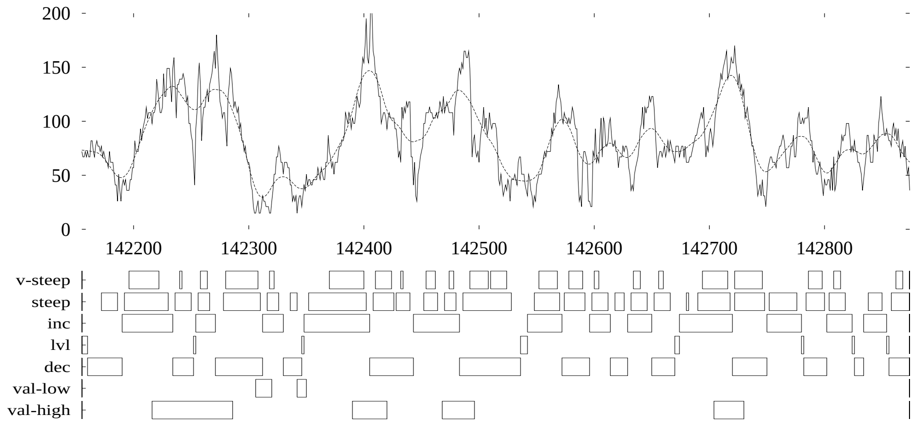


Fig. 5. Extracted features from time series: wind strength, Helgoland, April 1997.

The data has been measured hourly and we used 3, 6, and 9 years of data from 1991-1999 to test the algorithm. We have applied kernel smoothing in order to compensate for noise and to get more robust estimates of the first

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

and second derivative. Then, the smoothed series have been partitioned into primitive patterns like “increasing”, “concave”, “high-value”, etc. See Fig. 5 for an example.

Table 1 shows the performance of the pattern mining algorithm with different average state densities, window widths, and state series lengths. The threshold supp_{min} has been chosen to be 2% of the data period in all runs. The computation times ranged from a few seconds to 20 minutes on a 550 MHz Pentium III processor with 256MB main memory. We can see that the pruning techniques were quite efficient, besides a few exceptions, only 1-3% of all processed patterns became candidate patterns. The artificially generated data set has a rich pattern structure, on the average 45% of all candidates became frequent patterns. This value increases if we consider only runs with large window widths. If the state density D (average number of state intervals visible in the sliding window) is fixed, the run time is roughly linear in the size of the state series.

(a)			(b) years '97-'99				(c) years '94-'99				(d) years '91-'99			
S	W	D	F	F/C	C/P	T	F	F/C	C/P	T	F	F/C	C/P	T
8	18	3.42	191	30.3	5.1	1.19	178	29.4	5.0	1.31	28	7.1	53.1	0.56
8	30	5.70	1,126	56.2	3.4	2.95	1,055	54.9	3.5	3.03	96	20.9	8.2	0.65
8	42	7.98	4,904	78.3	2.3	8.27	4,459	78.1	2.3	7.97	249	27.3	7.2	1.11
15	18	5.40	1,071	55.8	2.7	2.06	471	25.0	1.6	2.19	829	46.7	1.8	2.01
15	30	9.00	2,779	42.6	2.4	5.24	2,618	41.6	2.5	5.38	2,024	37.1	2.8	4.67
15	42	12.6	12,900	67.3	1.3	16.0	11,986	66.4	1.4	15.7	9,618	63.3	1.5	13.4
27	18	8.28	1,600	25.2	2.1	4.27	1,562	24.9	2.1	5.34	1,359	23.0	2.3	4.08
27	30	13.8	9,767	42.7	1.8	12.4	9,184	41.6	1.8	14.5	7,082	37.8	2.0	10.1
27	42	19.3	48,832	65.3	0.7	43.3	45,302	64.2	1.0	49.0	34,872	60.9	1.1	32.7

Table 1. Results of the algorithm. In all experiments the threshold supp_{min} has been chosen to be 2% of the time series length (3, 6, and 9 years). Column S denotes the number distinct states in the series, column W denotes the window width (hours), column D the average state series density (average number of states visible in the window). Column F contains the number of frequent patterns, F/C the percentage of frequent patterns among candidate patterns, and C/P the percentage of candidate patterns among processed patterns (that is, candidate and pruned patterns). Column T shows the run time per 1000 state intervals in the series.

Subpattern tests have to be performed for all potential candidates whenever the content of the sliding window changes. Testing an n-window for a k-pattern is $O(n \cdot k^2)$ if all states are distinct, but may get much more complex the more identical states occur. Due to the complexity of the temporal patterns, the check requires some backtracking mechanism, which is computationally expensive. However, the temporal patterns are quite restrictive (even redundant), preventing us from too much backtracking. Another point is the number of “uninteresting” associations that are generated by the interval extraction: If the state series represents extracted local trends in time series it is natural that we observe many frequent patterns like “increasing segment before decreasing segment” etc. These uninteresting frequent patterns can be combined to patterns with more than 2 states arbitrarily and have considerable impact on the number of frequent patterns (and thus on run-time).

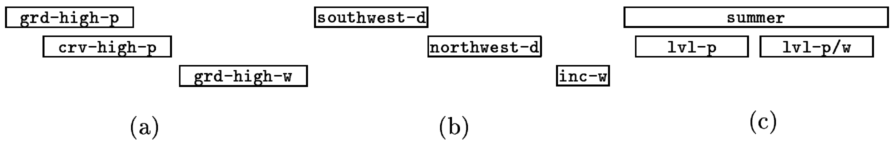


Fig. 6. Some exemplary rules. The bars indicate the temporal relationship between the intervals, their length has been chosen arbitrarily. The label in the bar describes a condition that holds in the interval (where `grd` denotes gradient, `crv` curvature, etc.).

Due to lack of space, here are only some exemplary rules. We have generated only those rules with a conclusion lying in the future (with respect to the intervals in the premise). Among them, there were many rules predicting a high gradient in wind strength, Fig. 6(a) shows one of them: If a period of highly increasing or decreasing air pressure overlaps a period of high curvature, it is very likely that the wind strength will change quickly (with a high gradient). The depicted rule occurs also with *during* and *meets* relationships between the air pressure states, so a disjunctive combination as described in Sect. 5.4 is promising. Figure 6(b) is an example for a rule that concludes from a change in wind direction to a strong change in wind strength. The rule in Fig. 6(c) tells us that stable weather (air pressure is nearly constant) is likely to be continued in summer, that is, a constant air pressure segment is followed by another constant air pressure period with low winds. Similar rules for other seasons can also be found, but with a much lower confidence value.

On the average, the confidence values of the rules are comparatively low (about 40-60% for the examples). This is because simple patterns (used in the premise) can be observed longer than complex patterns (patterns comprising premise and conclusion). To illustrate this, review Fig. 3(a)-(b), where the support of pattern “A” is greater than the support of pattern “A *overlaps* B”, although A has the same length in both cases. This leads to confidence values below 1 even if every A overlaps a B in the examined state series. We are investigating on other measures for rule evaluation in [9].

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 Sect. 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 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 if the known rules incorporate more variables than readily available. For instance, weather forecasting rules as discussed by Karnetzki [10] 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. Selection of the best rules gets further treatment in [9].

Acknowledgements. The author would like to thank Prof. F. Klawonn for critical review of this paper and the Deutsche Wetterdienst for providing the data.

References

[1] R. Agrawal, K.-L. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proc. of the 21st Int. Conf. on Very Large Databases*, Zürich, Switzerland, 1995.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In [8], chapter 12, pages 307–328. MIT Press, 1996.

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

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

[5] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In [8], chapter 9, pages 229–248. MIT Press, 1996.

[6] A. C. Capelo, L. Ironi, and S. Tentoni. Automated mathematical modelling from experimental data: An application to material science. *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, 28(3):356–370, Aug. 1998.

[7] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. 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.

[8] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

[9] F. Höppner and F. Klawonn. Finding informative rules in interval sequences. In *Proc. of the 4th Int. Symp. on Intelligent Data Analysis*, volume 2189 of *LNCS*, pages 123–132, Lissabon, Portugal, Sept. 2001. Springer.

[10] D. Karnetzki. *Luftdruck und Wetter.* Delius Klasing, 3rd edition, 1999.

[11] K. B. Konstantinov and T. Yoshida. Real-time qualitative analysis of the temporal shapes of (bio)process variables. *AIChE Journal: Chemical Engineering Research and Development*, 38(11):1703–1715, Nov. 1992.

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

[13] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Technical Report 15, University of Helsinki, Finland, Feb. 1997.

[14] S. A. McIlraith. Qualitative data modeling: application of a mechanism for interpreting graphical data. *Computational Intelligence (Theory and Practice)*, 5:111–120, 1989.

[15] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the 5th Int. Conf. on Extending Database Technology*, pages 3–17, Avignon, France, Mar. 1996.