A Temporal Representation of Point-Interval Relations including Calendar Events

Hanna Bauerdick, Björn Gottfried

Artificial Intelligence Group
Centre for Computing Technologies
Universität Bremen
{hbauerdick|bg}@tzi.de

Abstract. In this paper we shall discuss knowledge discovery problems arising in time series. A relation algebra is applied for the purpose of representing both dependencies among single items and dependencies between single items and intervals. This enables one to specify and recognise even complex interrelationships among items in data streams. In addition, intervals like calendar events can be incorporated into the patterns. An example is set for the case of thrown alarms in communication networks[1].

Keywords: Temporal Patterns, Calendar Events, Temporal Reasoning

1 Introduction

The representation of relationships in temporal data mining approaches is a big issue and has great impact on the expressiveness of patterns [10]. In several domains such expressive patterns are of great use, e.g. for the discovery of alarm patterns in the context of fault management, namely to improve the understanding of the network and in order to generate fault prediction rules; but also for such diverse purposes as for the identification of proteins on the basis of amino acids and for the prediction of financial and stock markets. Commonly, a temporal order is imposed on these data streams and we are normally concerned with long sequences of events. The temporal information of one event is often composed of a date and a time of day. In particular, date and time can be used to associate events to certain calendar events, such as days, weeks, months or even public holidays. These calendar events can considerably increase the diversity of detected patterns, even if nothing else is known about the events. In addition to calendar events, temporal data mining approaches normally contain a sliding time window which is scanned for new patterns (1) in order to reduce the complexity of the search space and (2) in order to include the fact that patterns are generally more reasonable if they include events which are temporally

[1] Funded by the European Commission, through IST project MDS, contract no. 26459.

© Revue MODULAD, 2007

close enough. Therefore, we have three sorts of objects among which relationships are to be considered: points, intervals and time windows (which can also be represented as intervals).

1.1 Related Work on Temporal Points and Intervals

These three sorts of objects can also be found in the related work on temporal data mining [10]. Several approaches deal with sequences of *point-like events*, which have to be analysed in order to find temporal patterns in the data. Mannila et al. [11] look for frequent temporal patterns in alarm sequences of networks. Here, an event refers to a point in time and to a given event type. Their goal is to find temporal relations between events. However, they only define two possible relations between events: serial and parallel. If two events persist in a serial relation, there exists a total order between them (like event A happened before event B). In a parallel relation the events have no order at all. These two temporal relations are also used in several other approaches (e.g. in [3]).

Bettini et al. [2] incorporate what they refer to as multiple time granularities: months, weeks, business day; they model them as *intervals* in the event correlation process. They relate occurred events to these granularities. For this purpose, a *contains* relation is defined between time points and granularity intervals. This same relation is used to define a hierarchical structure between time granularities. Consequently, patterns like 'A and B occurred on the same business day' can be detected. However, a hierarchical order between the intervals cannot always be found. E.g. if one week occurs between two months, Bettini et al. specify the *contains* relation as undefined. Other relations are not considered in their approach.

In order to reduce the complexity of their approach Mannila et al. [11] introduce a sliding *time window* which is then scanned for temporal patterns. They argue that the size of the time window should be user-defined. Since the quality of the approach highly depends on the chosen window size, it is quite challenging to find an adequate time window size. Alternatively, Casas-Garriga [3] suggest an algorithm which automatically adjusts the window width of a pattern according to the number of elements of the pattern. Further approaches exist. However, those which are particular closely related to our own one are referenced below when we introduce our representation.

1.2 Overview

By contrast to existing methods, our approach aims at including — at least on the level of ordering information — all possible temporal relations between points themselves and points and intervals. For this purpose, we introduce a relational system which allows relations systematically to be dealt with. For this purpose, a number of jointly exhaustive and pairwise disjoint point-point and point-interval relations are used. By this means, problems arising about undefined relations are avoided [2]. More important, these relations form the basis of a relation algebra that can be employed by using standard constraint satisfaction algorithms; this

© Revue MODULAD, 2007

enables one to solve recognition and relaxation problems as well as consistency checking tasks.

The remaining paper is structured as follows: A short description of the general problem situation and an introduction to fault management, as an example domain, is given in Section 2. Our own approach is introduced in Section 3. Section 4 demonstrates the applicability of our representation and explains a pattern recognition algorithm on the basis of an example scenario from the fault management domain. The paper concludes with Section 5 and provides an overview of future work.

2 Problem Description

Several domains have to deal with large amounts of temporal data, e.g. in fraud detection, patterns that indicate a case of fraud, and in fault management, sources of faults are to be identified; for the purpose of making predictions about stock markets complex relationships among shares, exchange rates and the like are of interest. Then, the goal consists in finding either frequent patterns or specific patterns among objects in temporal data streams.

2.1 Points and Intervals

In this section the general problem situation is described and subsequently motivated by an example domain: the fault management domain. It shows that there are three sorts of objects which are of interest to us: point-like events (the alarm of a device), expanded fractions of time (the days of the week), and specific periods of time (which cluster all events that might relate to each other). Reconciling points in time and expanded time segments so as to allow relations among both kinds of objects to be defined, is what we will finally aim at.

Time Points: In quite a few application domains, like fraud detection, supermarket sales information and fault management, long sequences of time point events have to be analysed and checked for temporal patterns, i.e. we are concerned with points in time (instead of time segments). Such an event normally consists of an attribute vector which further describes and specifies the occurred event. Here, however, we are interested in the temporal dimension and consider, as usual, a *time stamp*, which consists of date and time. For instance we describe, that

event B *follows* A

Time Intervals: In order to include how point-like events relate to expanded events, an additional level is required that incorporates intervals. Then, patterns arise which relate time points to intervals. The main example for this kind of relations is how an event relates to the calendar. Different granularities like days

© Revue MODULAD, 2007

of the week, weekends, months, etc. can then be described, as can concepts such as holiday time or bank holidays. Due to this layer of expanded events, more complex patterns can be specified, e.g.

event A is always followed by B *on Mondays*

Time Windows: In the spatial domain, Tobler's First Law of Geography holds: *Everything is related to everything else, but near things are more related than those far apart* [12]. We adopt this notion to the temporal domain, and for the purpose of specifying patterns in the temporal domain we confine relations among events which are temporally close enough. This closeness is usually defined by time windows, which constrain the influence of events depending on how far apart they are:

event A is always followed by B on the *next* Monday

In sum, we deal with points and intervals, i.e. with point-like events and expanded events, and we are interested in how they relate:

point-like events relative to each other, or
point-like events relative to expanded events.

Expanded events are either specific (calendar) events or time windows. Whereas the former events are precisely defined, the latter represents in a variable way the temporal closeness of point events, i.e. whether point events at a particular time can influence other point events at another time.

2.2 The Fault Management Domain

After having outlined the situation we are faced with, we shall now motivate this approach on the basis of the fault management area. Generally, fault management is divided into three different tasks: fault detection, fault localisation and fault correction (cp. [7, 8]). While fault detection is concerned with the recognition of failures, fault localisation approaches aim at identifying root causes of failures. The latter, eventually, are eliminated by the failure correction component of the fault management system.

The required input for most fault management components are (point-like) alarm events, which are generated by a network resource if its behaviour deviates from normality (i.e. the network is event-driven). These events are collected in a central component and then analysed in order to detect and localise the root cause. Alarm events consist of several attributes which characterise the detected failure. Since information about failures are provided by network resources, they highly depend on their vendors. Consequently, the attributes themselves may differ from network to network, and even within single networks, value ranges of attributes vary from subnet to subnet. Thus, the unification of attributes is often a big problem in the fault management area.

© Revue MODULAD, 2007

Fault localisation alone, is already a challenging issue. One aspect of fault localisation is the recognition of alarm patterns which frequently occur together in time; it is assumed that they result from the same root cause. To support the localisation of a fault the grouping of alarms according to their probable root cause and the generation of high level alarms would be of much use to the network operator [5]. In addition, one further step might be the generation of alarm prediction rules, which could be used to identify faults earlier and thus take early precautions to prevent severe effects of these faults from happening. In any case, temporal patterns of events are to be described. For the purpose of finding adequate ways for describing these patterns, we apply the trichotomy introduced in the previous paragraph to the fault management domain, as follows:

Time Points: From the temporal point of view, a network event is regarded as a point in time. Relations among network events are consequently relations between time points.

Time Intervals: Network events are to be related to specific calendar events, because they frequently determine the traffic in a network: on a Sunday there could be less traffic than on a Monday; and in the morning there is a traffic burst, while at noontide the traffic calms down. From what follows, network events are to be related to calendar events; in other words, points in time are to be related to intervals in time.

Time Windows: Network alarms which are closely related in time, have a higher probability that they arise due to the same root cause. Consequently, adequate time windows should be taken into account.

3 The Representation

The previous section introduced the fault management domain as an application example for temporal pattern discovery. This section concentrates on our own approach for the representation of temporal relations. In addition, an algebra is introduced which allows to reason about these relations.

3.1 Temporal Patterns as Relations

We are faced with two kinds of objects, points (representing points in time) and intervals (representing events with a duration). For two points in time, three relations exist: before ($<$), equal ($=$), and after ($>$) [13]. In the case of intervals, we distinguish thirteen relations [1]. We want to consider both point-point relations as well as point-interval (and conversely interval-point) relations. Interval-interval relations are of no interest for us. Intervals in our case represent calendar events, i.e. events with a duration. We do not want to describe relations

© Revue MODULAD, 2007

Table 1. Eleven relations exist which relate single points (P_i) to other points or to intervals (I_k).

Name	Symbol	Converse	Relation
before	$<$	$>$	$\{P_i, I_k\} < \{P_j, I_l\}$
meets	m	mi	$\{P_i, I_k\}$ m $\{I_k\}$
starts	s	si	$\{P_i\}$ s $\{I_k\}$
started by	si	s	$\{I_k\}$ si $\{P_i\}$
equal	$=$	$=$	$\{P_i\} = \{P_j\}$
during	d	c	$\{P_i\}$ d $\{I_k\}$
contains	c	d	$\{I_k\}$ c $\{P_i\}$
finishes	f	fi	$\{P_i\}$ f $\{I_k\}$
finished by	fi	f	$\{I_k\}$ fi $\{P_i\}$
met by	mi	m	$\{P_i, I_k\}$ mi $\{I_k\}$
after	$>$	$<$	$\{P_i, I_k\} > \{P_j, I_l\}$

between calendar events, but how either point-like events relate to each other or how they relate to calendar events.

As a consequence, the following relations can occur between single point-objects and single interval-objects: before ($<$), meets (m), starts (s), during (d), and finishes (f). Considering two points, it might also be possible that they coincide, in which case they are in relation equal ($=$). Additionally, we have to consider the converse relations which are listed in Table 1. By contrast to Allen's approach we have no overlap relation, since two points either coincide or they are disjoint; similarly, a point either meets an interval, or it coincides with its start-point. An overlap relation between a point and an interval is not realisable.

Other authors have been suggested to consider fewer point-interval relations [6, 4], namely $<$, s, si, d, c, f, fi, and $>$, i.e. they go without meets-relations. Since we will partition the time domain into discrete slices, it makes absolutely sense to distinguish a meets relation from, for example, the before relation. Also, point-point and point-interval relations are generally dealt with separately. Here, we need to integrate these systems and consider in particular $<$ and $>$ as to be equal regardless of whether their arguments are points, intervals, or a mix of both.

However, in the context of our domain it makes sense to reduce the proposed relations by assigning m and mi to $<$ and $>$, respectively. Furthermore, we assign both s and f to d. As a consequence, we finally arrive at a number of five relations, which are depicted in Table 2.

3.2 Reasoning about Temporal Patterns

For the purpose of reasoning about the relations just introduced, i.e. reasoning about temporal patterns, we define a relation algebra according to [9]. It consists of nine elements:

> the universe, $\mathcal{M}$, which consists of the atomic patterns (the five basic relations);

© Revue MODULAD, 2007

Table 2. After assigning m to $<$ and the starts and finishes relations to d, five relations finally remain.

Name	Symbol	Converse	Relation
before	$<$	$>$	$\{P_i, I_k\} < \{P_j, I_l\}$
equal	$-$	$-$	$\{P_i\} - \{P_j\}$
during	d	c	$\{P_i\}$ d $\{I_k\}$
contains	c	d	$\{I_k\}$ c $\{P_i\}$
after	$>$	$<$	$\{P_i, I_k\} > \{P_j, I_l\}$

	$<$	c	$=$	d	$>$
$<$	$<$	$<$	$<$	$< \mathsf{d}$	$< \mathsf{d} \cup \mathsf{c} >$
c	$< \mathsf{c}$	c	c	$\mathsf{d} = \mathsf{c}$	$\mathsf{c} >$
$=$	$<$	c	$=$	d	$>$
d	$<$	$< \mathsf{d} = \mathsf{c} >$	d	d	$>$
$>$	$< \mathsf{d} = \mathsf{c} >$	$>$	$>$	$\mathsf{d} >$	$>$

Fig. 1. The composition table.

- the identity relation, $=$, shows the temporal identity of two events;
- the $\emptyset$ designates the impossible pattern;
- the universal relation, U, designates the pattern which holds in any case (i.e. it is a set of all possibilities);
- the unary operation $\bar{}$ shows the complement of a pattern;
- the unary operation converseness, $\breve{}$, shows the pattern which arises when the ordering of objects is exchanged;
- the binary operation composition, $\circ$, computes the transitivity for a binary pattern;
- the binary operation intersection, $\cap$, extracts common parts of two patterns;
- the binary operation union, $\cup$, merges together two patterns;

The five operations are defined in the following way. In doing so, for the converse operation Table 1 is used and for the composition operation the composition table in Figure 1 is employed.

$$\overline{\mathsf{m}_i} - \{(\mathsf{O},\mathsf{P}) | (\mathsf{O},\mathsf{P}) \notin \mathsf{m}_i\} \tag{1}$$

$$\breve{\mathsf{m}_i} = \{(\mathsf{O},\mathsf{P}) | (\mathsf{P},\mathsf{O}) \in \mathsf{m}_i\} \tag{2}$$

$$\mathsf{m}_i \circ \mathsf{m}_j = \{(\mathsf{O},\mathsf{Q}) | \exists \mathsf{P} : (\mathsf{O},\mathsf{P}) \subset \mathsf{m}_i \wedge (\mathsf{P},\mathsf{Q}) \subset \mathsf{m}_j\} \tag{3}$$

$$\mathsf{m}_i \cap \mathsf{m}_j - \{(\mathsf{O},\mathsf{P}) | (\mathsf{O},\mathsf{P}) \in \mathsf{m}_i \wedge (\mathsf{O},\mathsf{P}) \in \mathsf{m}_j\} \tag{4}$$

$$\mathsf{m}_i \cup \mathsf{m}_j - \{(\mathsf{O},\mathsf{P}) | (\mathsf{O},\mathsf{P}) \in \mathsf{m}_i \vee (\mathsf{O},\mathsf{P}) \in \mathsf{m}_j\} \tag{5}$$

There exist two problem classes which can be managed by the proposed approach: configuration problems and consistency problems. The first one considers the definition of specific configurations, which, for instance, describe particular

© Revue MODULAD, 2007

system states; the second one is about the maintenance of a valid system state. In the former case we are in need of methods which systematically allow us to browse through the configuration space of valid patterns; in the latter case we need to check whether all system constraints (that together indicate a valid system state) are simultaneously satisfied.

Here, the temporal data stream of network events indirectly informs us about our system, i.e. whether specific patterns occur which, for instance, show us that a specific network service is currently used. On the other hand, the patterns can be employed for the purpose of checking whether our system is currently in a valid state. Moreover, the network operator might specify specific patterns which indicate particular problems he wants to get informed about, or he specifies conditions which indicate consistent network configurations. Then, as soon as an inconsistency arises a constraint is violated and the network operator gets informed.

3.3 Network Consistency

Given a set of temporal patterns $\mathbf{M}$, we want to know whether $\mathbf{M}$ is consistent. Being faced with a relation algebra we can regard this consistency problem as a constraint satisfaction problem (CSP) and can solve it by using standard methods. The computational evaluation of the consistency of a constraint net is performed as follows. In order to achieve arc-consistency

$$\forall_{\mathsf{O},\mathsf{P}} : \mathsf{Or}'\mathsf{P} :\!- \mathsf{Or}\mathsf{P} \sqcap \breve{\mathsf{Pr}}\mathsf{O}, \mathsf{r}, \mathsf{r}' \in \mathcal{P}(\mathcal{M}) \tag{6}$$

and in order to achieve path-consistency

$$\forall_{\mathsf{O},\mathsf{P},\mathsf{Q}} : \mathsf{Or}'\mathsf{P} :\!- \mathsf{Or}_1\mathsf{P} \sqcap (\mathsf{Or}_2\mathsf{Q} \circ \mathsf{Qr}_3\mathsf{P}), \mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3, \mathsf{r}' \in \mathcal{P}(\mathcal{M}) \tag{7}$$

These two steps are to be performed until no new relations are inferred. As the empty relation denotes an inconsistent scenario, as soon as the empty relation is deduced, the constraint net will have been proven to be inconsistent because any empty relation will remain empty under any further computations of equations 6 and 7. When the network has stabilised without inferring the empty relation the constraint net has been shown to be path-consistent.

For instance, we are concerned with four events, e_1, e_2, c_1, and c_2, and a number of three constraints among them: $\mathbf{M} = \{\mathsf{e}_1 < \mathsf{c}_2, \mathsf{e}_1 < \mathsf{c}_1, \mathsf{e}_2\ \mathsf{d}\ \mathsf{c}_1\}$. The graphical representation of this constraint network shows the given constraints. Some relations are not constrained, e. g. between e_1 and e_2 which is indicated by arrows without labels; consequently, the universal relation holds between those events. Furthermore, the identity relation holds for the relation between each event and itself, and we obtain the constraint network which is shown on the left hand side of Figure 2. After having applied equation 6, we obtain an arc-consistent network in which more relations are constrained (the middle part of Figure 2). For the purpose of achieving a path-consistent network, we apply equation 7 and obtain the rightmost part in Figure 2. For instance, the relation

© Revue MODULAD, 2007

r between e_1 and e_2 is computed as follows:

$$e_1 \; r \; e_2 = e_1 \; r_1 \; c_1 \;\circ\; c_1 \; r_2 \; e_2 = e_1 < c_1 \circ c_1 \; c \; e_2 = e_1 < e_2 \qquad (8)$$

As we then have derived other relations, we now have again employ equation 6 and obtain the next network. Impossible relations are deleted from e_2 r c_2, i.e. a calendar event cannot relate to an event by = or d. Eventually, it turns out that the network is consistent since the empty relation could not be inferred.

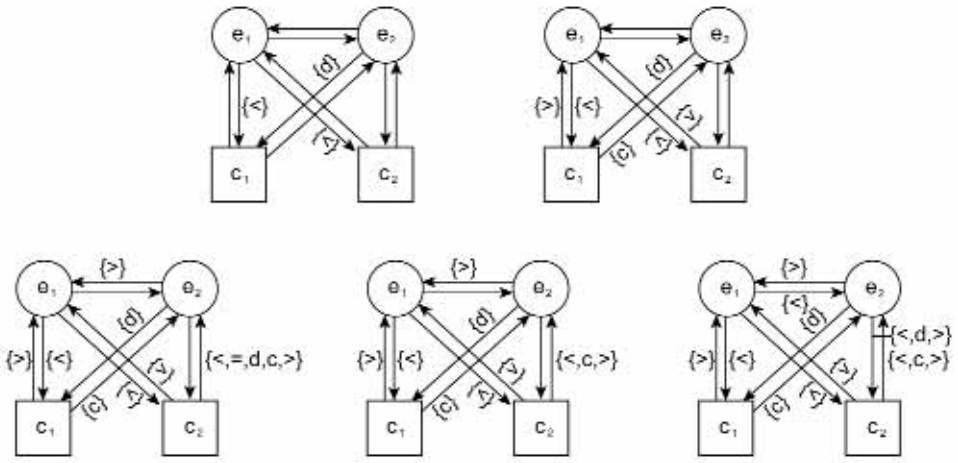


Fig. 2. Consistency Check Using Constraint Propagation

4 Example Scenario

This section explains the pattern representation and recognition approach in detail by means of an example scenario in the context of fault management. The first section 4.1 describes the initial situation of the scenario. Subsequently, an example pattern is shown to demonstrate the applicability of our approach (cp. Section 4.2). Section 4.3 presents the pattern recognition algorithm by example.

4.1 Customer Complaints in Fault Management Systems

Customer complaints are common when dealing with fault management issues. They occur, if not all faults can be found at an early stage and if they have been overlooked by the network operator or the fault management component. In such cases, the customers of the network discover a deviation in the behaviour and call the network operator to report the failure. These reports normally have to be processed by humans directly due to imprecise failure descriptions. However,

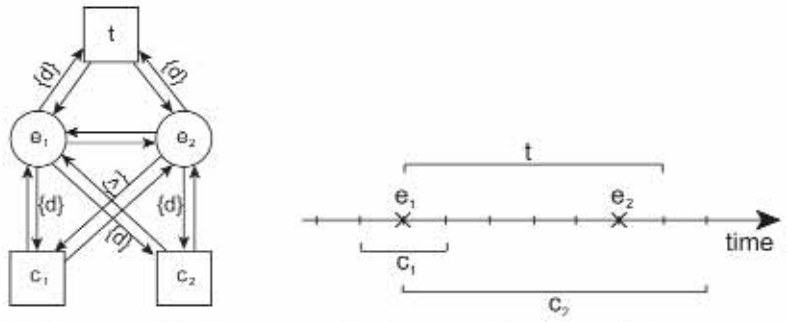
© Revue MODULAD, 2007

customer satisfaction is the greatest need of the network operator. Consequently, such situations should be avoided if possible. If the root cause of the failure is identified by the operator, one solution is to extract a pattern from the failure situation. This would help identifying the fault earlier next time.

The scenario for our approach will be exactly this situation. After receiving a customer complaint, the network operator identifies its root cause in the network and an event pattern for the situation is extracted. This pattern is saved in the pattern knowledge base of the fault management component. During the analysis, the temporal input data is scanned for patterns of the knowledge base. If a pattern is recognised, the network operator will be informed.

4.2 Representation of Patterns

Possible temporal relations within patterns are described in Section 3.1. These relations form a powerful language to specify temporal patterns. Here, the example of Section 3.3 is slightly modified and expanded to demonstrate how the trichotomy of different sorts of events is employed in a pattern (cp. Fig. 3). Within our scenario the network operator detects a situation similar to Fig. 3(b) and extracts the pattern described in Fig. 3(a). The letters of the time points represent the event types (in our case the types of alarms, e.g. signal loss, server is down, etc.). Due to simplicity reasons, the event types, calendar intervals and time windows are abbreviated in the following example.



(a) Constraint Graph (b) Scenario Scheme for the Pattern

Fig. 3. Example Pattern with Time Window

The pattern presented in Fig. 3 contains slight modifications compared to the pattern example described in the consistency section (cp. Section 3.1). First, the time window t and consequently its relations to both time point events are added to the pattern. Secondly, the original relations of Fig. 2 have been modified. Now, a number of six temporal constraints exist among the events and intervals, which are all depicted in Fig. 3(a): $\mathbf{M} = \{e_1\ \mathsf{d}\ c_1, e_1\ \mathsf{d}\ c_2, e_2 > c_1, e_2\ \mathsf{d}\ c_2, e_1\ \mathsf{d}\ t, e_2\ \mathsf{d}\ t\}$.

© Revue MODULAD, 2007

Fig. 3(b) visualises the temporal relations between the events and intervals and gives a scenario scheme for the pattern. All events and intervals are arranged on a timeline according to the temporal constraints.

The time window t plays a special role in the constraint graph. In a sense, it does not belong to the actual pattern. Rather, only patterns within single specific time windows are detected; i.e. during the recognition phase the events have to occur within the same time window, in order to get detected as a pattern.

4.3 Pattern Recognition

Pattern recognition corresponds to the process of detecting known patterns in the input stream of alarm events. In our approach, the patterns are represented using the temporal relations between points themselves and points and intervals. The time window is used to limit the scope of the pattern search. It *slides* over the data stream. In every step the window's content is scanned for already known patterns. Fig. 4 depicts one example scenario which contains the time window t, several alarm events (e.g. e_1 and e_5) and intervals (e.g. c_1 and c_2).

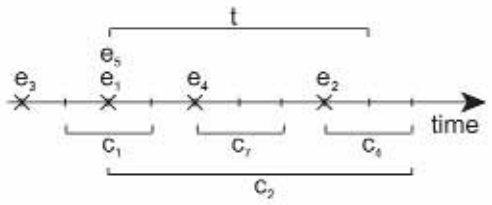


Fig. 4. Example Scenario for Pattern of Fig. 3

The composition table in Fig. 1 shows that when only measuring some relations (e.g. $<$, d and $=$), all other relations can be inferred by the same algorithm. Therefore, only the relations $<$, d and $=$ have to be detected in each time window. On this basis the following steps form an instruction list for the recognition of patterns. An adequate pattern recognition algorithm can be derived from these steps.

1. shifting the time window;
2. analysing the situation and deriving the relations $<$, d and $=$.
3. inferring the remaining relations.
4. comparing the pattern graph with the time window graph.

Taking the scenario of Fig. 4 as an example, the left subgraph of Fig. 5 can be extracted from the data within the second step of the recognition process. Likewise during consistency checking, the remaining relations are inferred from the known ones. As one can see, the right subgraph meets all temporal constraints from the pattern of Fig. 3(a). It is a subgraph of the pattern constraint graph.

© Revue MODULAD, 2007

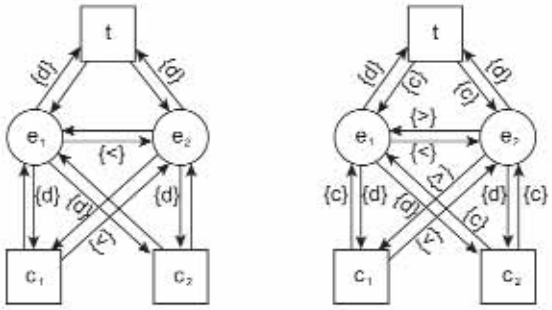


Fig. 5. Extract of Corresponding Constraint Graph

Following our scenario, after the recognition of the pattern the network operator will be informed about the detected failure. Now, correction actions can be executed early enough to avoid customer complaints connected with this failure.

5 Conclusion and Outlook

The algorithmic discovery and recognition of temporal patterns is still a challenging problem. Many domains, like fraud detection and fault management, have to rely on such algorithms due to the huge amount and the complexity of the data of today's applications. Within this paper, we proposed a representation for temporal patterns, including both point-like events and intervals, representing single events and calendar events or time windows, respectively. These patterns are represented using temporal constraints. Using common CSP (Constraint Satisfaction Problem) approaches, even the consistency of such patterns can be checked. In addition, an approach for the recognition of known patterns in the data stream has been introduced. The applicability of the representation and the recognition approach has been demonstrated using examples from the fault management domain.

Some open problems, however, remain. Our future work will address the introduction of an efficient approach for the discovery of patterns with our temporal relations based on the consideration of frequency distributions. In addition, some more complex examples from the fault management domain will be taken to further evaluate the method.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.
2. C. Bettini, X. S. Wang, S. Jajodia, and J.-L. Lin. Discovering frequent event patterns with multiple granularities in time sequences. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):222–237, 1998.

© Revue MODULAD, 2007

3. G. Casas-Garriga. Discovering unbounded episodes in sequential data. In *PKDD*, pages 83 94, 2003.
4. F. Coenen, B. Beattie, B. M. Diaz, T. J. M. Bench-Capon, and M. J. R. Shave. Temporal reasoning using tesseral addressing: towards an intelligent environmental impact assessment system. *Knowl.-Based Syst.*, 9(5):287–300, 1996.
5. F. Fessant, F. Clérot, and C. Dousson. Mining of an alarm log to improve the discovery of frequent patterns. In *Industrial Conference on Data Mining*, pages 144–152, 2004.
6. B. Han and A. Lavic. A framework for resolution of time in natural language. *ACM Trans. Asian Lang. Inf. Process.*, 3(1):11 32, 2004.
7. ITU. Information technology - Open Systems Interconnection - systems management: Alarm reporting function, recommendation X.733, 1992.
8. ITU. Management framework for Open Systems Interconnection (OSI) for CCITT applications, recommendation X.700, 1992.
9. P. Ladkin and R. Maddux. On binary constraint problems. *Journal of the Association for Computing Machinery*, 41(3):435 469, 1994.
10. S. Laxman and P. S. Sastry. A survey of temporal data mining. *Academy Proceedings in Engineering Sciences*, 31(2):173–198, 2006.
11. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259 289, 1997.
12. W. Tobler. A computer movie simulating urban growth in the Detroit region. *Economic Geography*, 46 (4):360–371, 1970.
13. M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. pages 377 382, 1986.

© Revue MODULAD, 2007