Fuzzy Temporal Relations for Fault Management

Hanna Bauerdick and **Björn Gottfried**
Artificial Intelligence Group
Center for Computing Technologies
Universität Bremen
{hbauerdick|bg}@tzi.de

Abstract

In this paper we shall introduce an approach that forms a basis for temporal data mining. A relation algebra is applied for the purpose of representing simultaneously dependencies among instants, dependencies between instants and intervals, and dependencies between intervals. This enables one to specify and recognise complex interrelationships among point events in data streams. An example is set for the case of thrown alarms in communication networks. This domain, however, suffers from uncertainties in temporal patterns. Therefore a fuzzification is applied to the proposed relational system[1].

Introduction

The representation of relationships in temporal data mining approaches is a big issue and has great impact on the expressiveness of patterns (Laxman & Sastry 2006). 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. In particular, in a number of domains (e. g. in fault management) it makes sense to analyse sequences of events in relation to certain calendar events, such as days, weeks, months, or public holidays. These calendar events, then, can considerably increase the diversity of detected patterns, even if nothing else is known about the events.

Related Work

Both sorts of objects instants and intervals are found in the related work on temporal data mining (Laxman & Sastry 2006). Several approaches deal with sequences of *point-like events*, which have to be analysed in order to find temporal patterns in the data. (Mannila, Toivonen, & Verkamo 1997) 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 (Casas-Garriga 2003).

Bettini *et al.* 1998, incorporate what they refer to as multiple time granularities: months, weeks, and business days; they model them as *intervals* in the event correlation process. Occurred events are in turn be related to these granularities. For this purpose, a *contains* relation is defined between instants and such 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. Other relations are not considered in their approach. Further methods exist, however; those which are particular closely related to our own one are referenced below when we introduce our representation.

Aim

By contrast to existing methods, our approach aims at integrating on the level of ordering information all possible temporal relations between instants and intervals; though they will then be restricted to what is relevant regarding the fault management domain. Interval-interval relations, for instance, are of interest in this domain insofar containment relations among different types of calendar events can be considered. 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, point-interval, and interval-interval relations are used. By this means, problems arising about undefined relations are avoided (Bettini *et al.* 1998). More important, these relations form the basis of a relation algebra that can be employed by using standard constraint satisfaction algorithms; this enables one to solve recognition and relaxation problems as well as consistency checking tasks. Furthermore, a method for the fuzzification of such a relational system can be applied in order to deal with uncertainty.

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

Overview

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 the following section. Then, our own approach is introduced and fuzzified afterwards. The applicability of the new representation is demonstrated and explained on the basis of an example scenario from the fault management domain.

Problem Description

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 two sorts of objects which are of interest for us: point-like events (alarms of devices) and events having a duration (the days of the week). Reconciling instants and intervals, so as to allow relations among both kinds of objects to be defined, is what we will aim at.

Instants: 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. Such events normally consist of an attribute vector which further describes and specifies the occurred event. Here, however, we are only interested in the temporal dimension. We describe, for instance, that

event B *follows* event A

Intervals: In order to include how point-like events relate to events having a duration, an additional level is required that incorporates intervals. Then, patterns arise which relate instants to intervals. The main example for this kind of relations is how an event relates to the calendar. Different granularities like days of the week, weekends, months, or forenoon versus afternoon can then be described, as can concepts like holiday time or bank holidays. Due to this layer of intervals, more complex patterns can be specified:

event B is always followed by event A *on Mondays*

In summary, we deal with instants and intervals, i. e. with point-like events and those having a duration, and we are interested in how they relate:

- instants relative to each other,
- instants relative to intervals, and
- intervals relative to each other.

The Fault Management Domain

After having outlined the situation we are faced with, we 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. (ITU 1992)). While fault detection is concerned with the recognition of failures, fault localisation approaches aim at identifying root causes of failures. The latter 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.

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 (Fessant, Clérot, & Dousson 2004). 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 temporal relationships introduced in the previous paragraph to the fault management domain:

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

Time Intervals: Network events are to be related to specific calendar events, because they sometimes 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, instants are to be related to intervals in time.

By default every computer network contains a certain degree of uncertainty resulting from lost alarm events and non-synchronised clocks of the network resources. In addition, frequently occurring situations may have slightly different constellations of events, but still should be detected as the same pattern. Pattern discovery and recognition algorithms should therefore be capable of dealing with such kinds of uncertainties; so has the underlying representation itself. Distinguishing whether a specific relation holds or not is insufficient. We have to distinguish whether the relation holds, whether it is at least probable that it holds, and that it is impossible.

The Representation

The previous section introduced the fault management domain as an application example in which specific temporal patterns occur. This section introduces a new representation that represents those patterns in a relational system. In order to be able to deal with network uncertainties, this system will be fuzzified afterwards according to (Guesgen 2003).

Temporal Patterns as Relations

We are faced with two kinds of objects, points (point-like events) and intervals (events with a duration). For two points in time, three relations exist: before ($<$), equal ($=$), and after ($>$) (Vilain & Kautz 1986). In the case of intervals, we distinguish thirteen relations (Allen 1983). We want to consider

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	$>$	$<$	$\{P_i, I_k\} > \{P_j, I_l\}$

both point-point relations as well as point-interval (and conversely interval-point) relations. Interval-interval relations are only of interest in a rather restricted way: the week is fragmentised into days, and single days into phases such as morning, noon, afternoon, and evening — overlap relations are in their explicit existence of no interest for us; they will later on be implicity dealt with by using fuzzification techniques. $\mathcal{I}$ denotes the set of possible intervals, and $\mathcal{E}$ denotes the set of possible instants. Together they form the set of possible objects $\mathcal{O}$ (i. e. $\mathcal{O} = \mathcal{E} \cup \mathcal{I}$).

The following relations can occur between instants and intervals: 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, because the point either meets the interval, starts it, finishes it or occurs during it.

Other authors have been suggested to consider fewer point-interval relations (Han & Lavie 2004; Coenen *et al.* 1996), 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 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 and arrive at a number of five condensed relations. Note that concerning the domain at hand, the relations are jointly exhaustive and pairwise disjoint. They are depicted in Table 2 and as a neighbourhood graph in Fig. 1. The neighbourhood graph depicts on the one hand how an instant can be related either to an interval (cf. leftmost graph of Fig. 1) or to an instant (cf. centred graph of Fig. 1) and on the other hand how intervals themselves can be related (cf. the rightmost graph of Fig. 1). Consequently, the graph can be used for reasoning about temporal patterns of all kinds. As can be seen, the set of possible, condensed relations $\mathcal{R}$ is composed of point-point relations $\mathcal{R}_\mathcal{E} = \{<, =, >\}$, point-interval relations $\mathcal{R}_\mathcal{I} = \{<, \mathsf{d}, >, \mathsf{c}\}$ and interval-interval relations $\mathcal{R}_\mathcal{C} = \{<, \mathsf{d}, >, \mathsf{c}, =\}$.

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	$>$	$<$	$\{P_i, I_k\} > \{P_j, I_l\}$

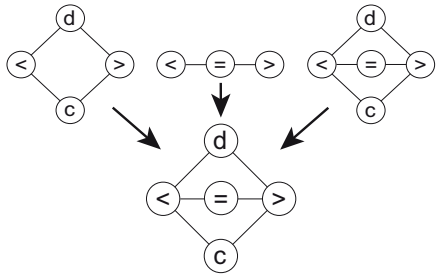


Figure 1: Conceptual neighbourhood graph of possible, condensed point-point & point-interval & interval-interval relations

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 (Ladkin & Maddux 1994). It consists of nine elements:

- the universe, $\mathcal{M}$, which consists of the atomic patterns (the five basic relations);
- 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 (a set of all possibilities);
- the unary operation, $^-$, is the complement;
- the unary operation converseness, $^\smile$, shows what arises when the ordering of objects is exchanged;

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

Figure 2: The composition table.

- 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 them together;

Assuming that o_1, o_2 and o_3 are of type $\mathcal{O}$, 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 Fig. 2 is employed.

$$\overline{m}_i = \{(o_1, o_2) | (o_1, o_2) \notin m_i\} \tag{1}$$

$$\breve{m}_i = \{(o_1, o_2) | (o_2, o_1) \in m_i\} \tag{2}$$

$$\begin{gathered} m_i \circ m_j = \\ \{(o_1, o_3) | \exists o_2 : \ (o_1, o_2) \in m_i \wedge (o_2, o_3) \in m_j\} \end{gathered} \tag{3}$$

$$m_i \cap m_j = \{(o_1, o_2) | (o_1, o_2) \in m_i \wedge (o_1, o_2) \in m_j\} \tag{4}$$

$$m_i \cup m_j = \{(o_1, o_2) | (o_1, o_2) \in m_i \vee (o_1, o_2) \in m_j\} \tag{5}$$

Fuzzifying Temporal Patterns

In many domains the data used for pattern discovery inhere a certain degree of uncertainty, for example in fault management the actual time of different network resources may vary and consequently the date of generated events. Thus, the arising uncertainty relates to fuzzy changes among neighbouring relations according to their conceptual neighbourhood relation (cf. Fig. 1). Therefore, we extend the formulation of temporal relations using fuzzy relations.

Fuzzifying Temporal Relations For clarity we proceed similar as (Guesgen 2003) and introduce fuzziness step by step. For this purpose we start by representing the crisp case alternatively. This representation will base on a characteristic function $\mu_r$which specifies a given temporal relation r as a function. Such a function has to be defined for each possible relation, r, as follows:

$$\mu_r : \ \mathcal{R} \longrightarrow \{0, 1\} \tag{6}$$

The characteristic function μ_r yields 1 iff the argument r' is identical with the relation r specified by the characteristic function:

$$\mu_r(r') = \begin{cases} 1, & \text{if r' = r} \\ 0, & \text{else} \end{cases} \tag{7}$$

Using this characteristic function the crisp relations introduced above can be described as sets of tuples. That is, for a relation r each tuple consists of one of the five relations and its value regarding the characteristic function of r. Consequently, each relation is represented as a set of five tuples. For example the relation $<$ between the two objects $o_1, o_2 \in \mathcal{O}$ is represented as follows:

$$\begin{gathered} o_1 \{(r, \mu_<(r)) | r \in \mathcal{R}\} o_2 \\ = o_1 \{(<, 1), (c, 0), (=, 0), (d, 0), (>, 0)\} o_2 \end{gathered} \tag{8}$$

By now, only the crisp case is covered, i. e. two cases are distinguished: relations are either accepted (i. e. $\mu_r(r') = 1$) or rejected (i. e. $\mu_r(r') = 0$). Fuzziness is introduced by assigning membership grades $\mu_{\tilde{r}}$ to relations. For this purpose the conceptual neighbourhood structure is used which reflects the conceptual similarity between relations. In case of $<$ we assign 1 to the membership grade if the given relation is $<$, α_1 if the relation is a neighbour of $<$, α_2 if the relation is a neighbour of a neighbour of $<$ and so on (with $0 \leq \ldots \leq \alpha_2 \leq \alpha_1 < 1$). Then, the fuzzy relation $\tilde{r}$ can be defined w. r. t. a given object o_1 (which is either an instant or an interval) by the following set of tuples:

$$\tilde{r}_{o_1}(o_2) = \{(r', \mu_{\tilde{r}, o_1}(r', o_2)) | r' \in \mathcal{R}\} \tag{9}$$

with

$$\mu_{\tilde{r}, o_1} : \mathcal{R} \times \mathcal{O} \longrightarrow [0, 1] \tag{10}$$

and

$$\mu_{\tilde{r}, o_1}(r', o_2) = \begin{cases} 1, & \text{if } r' = \tilde{r} \\ 0, & \text{if } (o_1, o_2 \in \mathcal{E} \wedge r' \notin \mathcal{R}_\mathcal{E}) \vee \\ & ((o_1 \in \mathcal{I} \vee o_2 \in \mathcal{I}) \wedge r' \notin \mathcal{R}_\mathcal{I}) \\ \alpha_i, & \text{else} \\ & \text{with } 0 \leq \alpha_i < 1, i = 1, 2, .. \end{cases} \tag{11}$$

α_i refers to the membership grade of r' w. r. t. $\tilde{r}$ (derived from the conceptual neighbourhood graph). At this i corresponds to the length of the shortest path between $\tilde{r}$ and r' in the neighbourhood graph. Using this representation the crisp relations can also be covered when $\alpha_1 = \alpha_2 = ... = 0$. Fig. 3 depicts the neighbourhood graph composed of the five basic relations and their corresponding membership grades w. r. t. the point-interval relation $>$. Note that in comparison to (Guesgen 2003) we have to distinguish whether we have to deal with two instants, one instant and an interval or two intervals, explaining the necessity to include o_1 in the expression $\mu_{\tilde{r}, o_1}(r', o_2)$.

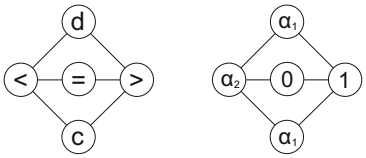


Figure 3: Possible point-point & point-interval relations and the membership grades w. r. t. the point-interval relation $>$

Including point-point, point-interval and interval-interval relations in a single neighbourhood graph will enable us to propagate constraints across all sorts of objects: Given that specific relations between point-point, point-interval and interval-interval relations hold. Then, as soon as some point-object is changed along the timeline we want to propagate the consequences across all kinds of relations, i. e. for all point-point and point-interval as well as for all interval-interval relations we want to derive the consequences — in particular how changes along point-point relations entail changes along point-interval relations.

Fuzzifying Temporal Reasoning In this section fuzzy compositions are discussed. Starting again with the crisp case, the composition of the two relations r_1 and r_2 is represented as follows when using the characteristic function:

$$\mu_{r_1 \circ r_2} \; : \; \mathcal{R} \; \longrightarrow \{0, 1\} \tag{12}$$

with

$$\mu_{r_1 \circ r_2}(r') = \begin{cases} 1, & \text{if } r' \in r_1 \circ r_2 \\ 0, & \text{else} \end{cases} \tag{13}$$

Similar as above the characteristic function is exchanged by a membership function in order to consider fuzziness. Consequently, the composition of two fuzzy relations $\tilde{r}_1$ and $\tilde{r}_2$ is based on membership grades. For the definition of $\mu_{\tilde{r}_1 \circ \tilde{r}_2}$ the min/max combination scheme from fuzzy set theory is used:

$$\mu_{\tilde{r}_1 \circ \tilde{r}_2, o_1, o_2} \; : \; \mathcal{R} \; \times \mathcal{O} \longrightarrow \{0, 1\} \tag{14}$$

with

$$\mu_{\tilde{r}_1 \circ \tilde{r}_2, o_1, o_2}(r', o_3) = \begin{cases} 0, & \text{if } (o_1, o_3 \in \mathcal{E} \wedge r' \notin \mathcal{R}_{\mathcal{E}}) \vee \\ & ((o_1 \in \mathcal{I} \vee o_3 \in \mathcal{I}) \wedge r' \notin \mathcal{R}_{\mathcal{I}}) \\ \nu_{\tilde{r}_1 \circ \tilde{r}_2, o_1, o_2}(r', o_3), & \text{else} \end{cases} \tag{15}$$

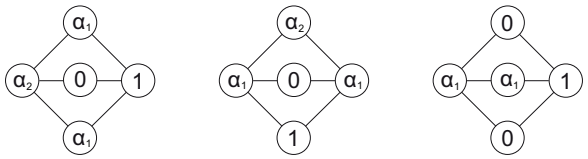
and

$$\nu_{\tilde{r}_1 \circ \tilde{r}_2, o_1, o_2}(r', o_3) = \max_{\forall r_3, r_4 \in \mathcal{R} : \mu_{r_3 \circ r_4}(r') = 1} \{\min\{\mu_{\tilde{r}_1}(r_3), \mu_{\tilde{r}_2}(r_4)\}\} \tag{16}$$

Each composition which results in r' is evaluated using the membership functions of $\tilde{r}_1$ and $\tilde{r}_2$ respectively. For each composition which results in r' the lowest membership grade of the composed relations is taken (the consequence can only be as strong as its weakest antecedent). This is done for every composition table entry which contains r' and the maximum of all these minimums is used (which is the highest membership value of the compositions). For example let us assume that $\tilde{r}_1$ is set to $>$, $\tilde{r}_2$ to c and r' is assigned with $<$. One possible composition which yields in $<$ is the composition $< \circ \mathsf{c}$. In this case the weakest antecedent is the relation $<$ which has a membership value $\mu_{>}$ of α_2. However, the best of all these compositions (i. e. the one with the best composed membership grade) is taken as the result. In our example, the membership value $\mu_{> \circ \tilde{\mathsf{c}}}(<)$ is α_1, because the best composition yields at that value. Fig. 4 depicts the membership graph of $>$, c and $> \circ \mathsf{c}$.

Example Scenario

This section explains the pattern representation by means of an example scenario in the context of fault management. The problem situation is described and it is argued why a fuzzy set representation of patterns is of great use within this domain. Subsequently, an example pattern is shown to demonstrate how our approach can be applied in the fault management domain.



(a) Relation $>$ (b) Relation c (c) Composition $> \circ \mathsf{c}$

Figure 4: Membership grades for composition $> \circ \mathsf{c}$ between an interval and an instant

Customer Complaints in Fault Management

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 network's 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, 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. Now the temporal input data can be scanned for this pattern to inform the network operator much earlier in the case of a fault; thus, many customer complaints will be avoided in the future.

Every computer network inheres a certain degree of uncertainty, which results from variations of the system time of the network resources and thus from the varying generation time of events. Also, similar situations in which patterns almost match should be captured, i. e. the temporal occurrence of events may also differ to a certain degree. Consequently, representing patterns using fuzzy temporal relations is quite reasonable in the fault management domain.

Representation of Patterns

Possible temporal relations within patterns and their fuzzification are described in the previous section. These fuzzy relations form a powerful language to specify temporal patterns. Here, a pattern is extracted from the problem situation depicted in Fig. 5 to show how to employ the fuzzy relations for pattern specification. It is also demonstrated how to reduce the complexity of the pattern recognition by only detecting a small set of relations and inferring the remaining ones.

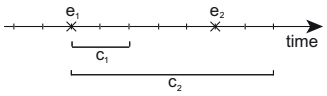


Figure 5: Scenario scheme for the example pattern

Let us assume that the network operator detects a situation similar to Fig. 5 and extracts the pattern described in Fig. 6(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 and calendar intervals are abbreviated in the following example.

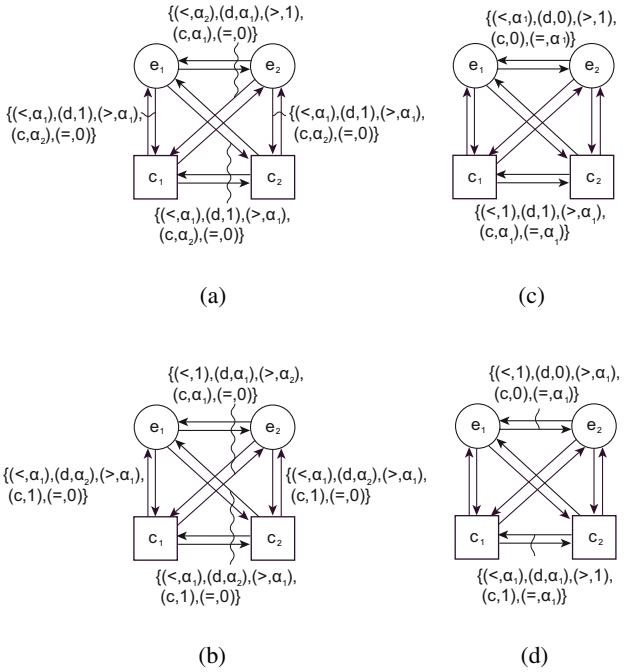


Figure 6: Constraint graph of an example pattern

In the situation depicted in Fig. 5 we are concerned with two instants (e_1 and e_2) and two intervals (c_1 and c_2). A number of four relations are specified among them. In the crisp case they would be represented as follows: $\mathcal{M} = \{e_1 \mathsf{d} c_1, e_1 \mathsf{d} c_2, e_2 > c_1, e_2 \mathsf{d} c_2\}$. When fuzzifying the relations, the representation of them would change as can be seen in Fig. 6(a). Here, the graphical representation of the pattern is shown as a constraint network. 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 each event and itself.

When defining a pattern not every relation has to be specified. Most of them can be inferred from others employing the composition operation. Even though some relations in the example pattern are not assigned to a fuzzy relation yet, all of them are indirectly constrained by already specified relations. Fig. 6 shows the constriction step-by-step. For simplicity reasons only the modified relations are depicted. As can be seen, it is inferred that instant e_1 is most likely followed by instant e_2. A general fuzzified CSP-algorithm can be found in (Guesgen 2003).

Conclusion and Outlook

The algorithmic recognition of temporal patterns is a challenging problem. Many domains like fraud detection and fault management have to rely on such algorithms due to huge amounts of data. Additionally these domains inhere a certain degree of uncertainty (e. g. in fault management due to lost alarms and unsynchronised clocks). We proposed a representation for temporal patterns which incorporates two different kinds of objects: instants and intervals. The fuzzification of the representation allows uncertain relations to be dealt with. The applicability of the fuzzy representation has been demonstrated using an example 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 fuzzy temporal relations based on the consideration of frequency distributions. In addition, we shall analyse how knowledge about the network topology will aid in improving results.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26:832–843.

Bettini, C.; Wang, X. S.; Jajodia, S.; and Lin, J.-L. 1998. Discovering frequent event patterns with multiple granularities in time sequences. *IEEE Transactions on Knowledge and Data Engineering* 10(2):222–237.

Casas-Garriga, G. 2003. Discovering unbounded episodes in sequential data. In *PKDD*, 83–94.

Coenen, F.; Beattie, B.; Diaz, B. M.; Bench-Capon, T. J. M.; and Shave, M. J. R. 1996. Temporal reasoning using tesseral addressing: towards an intelligent environmental impact assessment system. *Knowl.-Based Syst.* 9(5):287–300.

Fessant, F.; Clérot, F.; and Dousson, C. 2004. Mining of an alarm log to improve the discovery of frequent patterns. In *Industrial Conference on Data Mining*, 144–152.

Guesgen, H. W. 2003. When regions start to move. In *Proc. FLAIRS-03*, 465–469.

Han, B., and Lavie, A. 2004. A framework for resolution of time in natural language. *ACM Trans. Asian Lang. Inf. Process.* 3(1):11–32.

ITU. 1992. Management framework for Open Systems Interconnection (OSI) for CCITT applications, recommendation X.700.

Ladkin, P., and Maddux, R. 1994. On binary constraint problems. *Journal of the Association for Computing Machinery* 41(3):435–469.

Laxman, S., and Sastry, P. S. 2006. A survey of temporal data mining. *Academy Proceedings in Engineering Sciences* 31(2):173–198.

Mannila, H.; Toivonen, H.; and Verkamo, A. I. 1997. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.* 1(3):259–289.

Vilain, M., and Kautz, H. 1986. Constraint propagation algorithms for temporal reasoning. 377–382.