Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems

Christophe Dousson and Thắng Vu Dtfdng
Prance Telecom CNET, 2 avenue Pierre Marzin
F-22307 Lannion Cedex - Prance
{christophe.dousson, thang.vuduong}@cnet.francetelecom.fr

Abstract

We address the problem of knowledge acquisition for alarm correlation in a complex dynamic system like a telecommunications network. To reduce the amount of information coming from telecommunications equipment, one needs to preprocess the alarm stream and we propose here a way to acquire some knowledge to do that. The key idea is that only the frequent alarm sets are relevant for reducing the information stream: we aggregate frequent relevant information and suppress frequent noisy information. We propose algorithms for analysing alarm logs: first stage is to discover frequently occurring temporally-constrained alarm sets (called *chronicles)* and second stage is to filter them according to their interdependency level. We also show experimental results with an actual telecommunications ATM network.
Areas: knowledge acquisition, discovery, temporal reasoning, applications, monitoring.

1 Introduction

Telecommunications networks are growing in size and complexity, which means that a bigger and bigger volume of notifications needs to be handled by the management system. Most of this information is produced spontaneously by equipment (e.g., status change and dysfunction detection) and this message flow must be preprocessed to make effective management possible.

Filters based on a per-notification basis fail to perform an adequate information preprocessing required by human operators or by management application software, which are not able to process such amount of events. A preprocessing stage must decrease this information stream by suppressing superfluous notifications and/or by aggregating relevant ones.

Some papers deal with different approaches and propose more or less complex intelligent filtering: one can use some efficient rule-based languages (like in [Moller *et o/.*, 1995]), and/or object-based techniques like ECXpert (Event Correlation eXpert) which builds alarm correlation trees according to some handwritten rules [Nygate, 1995]. More specific techniques are also studied to capture time constraints between alarms [Dousson *et a/.*, 1993; Jakobson and Weissman, 1995].

In any case, the problem of expertise acquisition remains the same: how to feed the filtering system? Which aggregation rules are relevant? Since time information is apropos for the telecommunications alarm propagation, we also have to be able to deal with numerical time constraints. One way for such a knowledge acquisition is model-based approaches like [Bibas *et* al., 1995] or [Laborie and Krivine, 1997], which use models to build a simulator for generating relevant faulty scenarios.

Our approach is more akin to data mining techniques: it is based on a frequency analysis of actual alarm logs of the supervised system (the telecommunications network) to discover some frequent chronicles. The frequent chronicle approach is relevant to reach the aim of reducing the alarm stream: if a chronicle corresponds to a dysfunction, we aggregate the set of alarms for the human operator and if not, we only hide the corresponding alarms. At the moment, we do not use any extra knowledge, so the rule qualification (aggregation or suppression) must be performed by an expert.

The original idea stands in [Mannila *et a/.*, 1995] but their work focuses only on two types of chronicles (which are called episodes): the parallel ones (alarms are not ordered) and the serial ones (alarms are completely ordered). Some extensions for a more complex chronicle structure are proposed (but not tested) by combining serial and parallel chronicles. Moreover, only a user given upper bound of chronicle duration was allowed to be a time constraint. We extend our frequency analysis with a time representation, which is able to deal with numerical time constraints.

We present in this paper a system (called FACE - Frequency Analyser for Chronicle Extraction) that enables an expert to process some powerful analyses of alarm logs and to identify some recurrent phenomena. In Section 2, we expose some definitions for later use. Section 3 details the first stage of our log analyser, where frequently occurring chronicles in alarm logs are discovered; Section 4 corresponds to the second stage where a graph of dependencies between chronicles is established. Then in Section 5, before concluding, we show some results about an experiment with an actual ATM network.

2 Representation and Definitions

2.1 Time

We based our alarm correlation system on CRS, a Chronicle Recognition System similar to IxTeT[1], described in [Dousson, 1996], because of its real-time capabilities to deal with numerical time constraints.

For complexity reasons, this system relies on time-points as elementary primitives and considers the time as a linearly ordered set of discrete events. A time constraint between two time-points t_1 and t2 is represented by an interval, $[I^-, I^+]$, which corresponds to the lower and upper bounds on the temporal distance from t1 to t2. We also define a time constraint graph $\mathcal{T}$ as a set of time-points with time constraints between them (constraint between i and j is an interval denoted by $K_{\mathcal{T}}(i \rightarrow j)$). We define a partial order *(tighter than,* denoted by $\subseteq$) among constraint graphs as follows:

$$\mathcal{T} \subseteq \mathcal{T}' \equiv_{def} \forall (i,j) \in \mathcal{T}, K_{\mathcal{T}}(i \rightarrow j) \subseteq K_{\mathcal{T}'}(i \rightarrow j)$$

A constraint graph may have many equivalent representations. In particular, there is a unique equivalent constraint graph, which is *minimal* (with respect to $\subseteq$) and its computation (and its consistency check) is ensured by the well-known path consistency *Floyd-WarshaWs* algorithm with the complexity of $O(n^3)$ [Dechter *et al.*, 1991]. We do not allow disjunctive constraints since this problem becomes NP-hard.

In the following sections, we systematically apply this algorithm, and so, we always deal with the minimal time constraint graph.

2.2 Alarm

Alarm: an alarm is a pair *(A,t)* where *A* is the *alarm type* and *t* is the time instant (i.e., occurrence date) of the alarm.

Alarm occurrence: an alarm occurrence is a time-stamped alarm (e.g., *(A,* 4)).

Alarm log: an alarm log $\mathcal{L}$ is a time-ordered list of alarm occurrences. For example, *(A,* 1)(B3,3)(J4,4) (C,4)(A,7)(B,8)(C,9)(B,10)(B,12).

Due to time-point primitives, an alarm has no duration. If we need some, we can introduce two alarm types corresponding to the beginning and the end of an alarm. For instance, in the telecommunications management, one usually uses a LOS_{active} when the alarm *Loss Of Signal* appears, and a LOS_{clear} when it disappears.

We also suppose that alarms are correctly time-stamped in logs (which is often the case). The message propagation delay in a network must only be taken into account during the on-line recognition stage and this feature is actually performed by CRS.

[1]The main differences between them are the heuristics used during the recognition because CRS is specially developed for telecommunications networks, but it does not matter here because we use it as a black box.

2.3 Chronicle Model and Instance

Chronicle model: a chronicle model *C* is a pair (S, *T)* where *S* is a set of alarms and $\mathcal{T}$ is a constraint graph of their instants. We also denote $\mathcal{C}^N$ a chronicle with *N* alarms *(N* is called the *size* of $\mathcal{C}^N$).

For example, Figure 1 shows a breakdown link chronicle, where *LOS* stands for *Loss Of Signal* and *LOF* stands for *Loss Of Frame.*

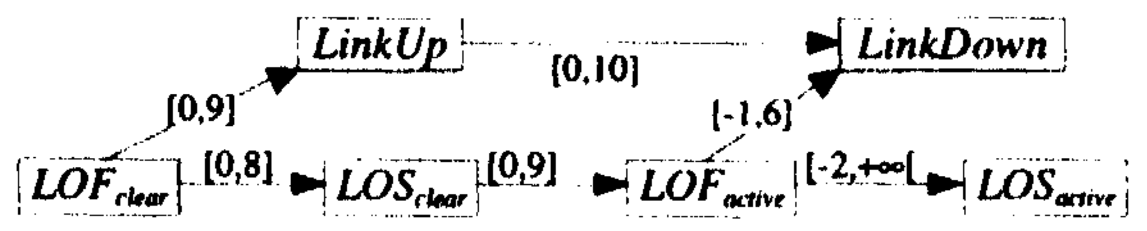

Figure 1: A breakdown link chronicle.

Unconstrained chronicle model: an unconstrained chronicle model (denoted by (S, •)) is a chronicle model with no time constraint and *no order* between alarms. We also use alarm types for the simplified notation of an unconstrained chronicle model as in the following example: *ABB* stands for $((A,t_1),(B,t_2),(B,t_3),.)$[2].

Chronicle instance: a chronicle instance *c* of a chronicle model *C* is a set of alarm occurrences which is consistent with the time constraints of *C.* For example, {(A, 1)(B, 3)(A, 7)} is an instance of the left chronicle of Figure 2.

Subchronicle instance: an instance c is a subchronicle instance of an instance *c' iff* c is a subset of C' (e.g., {(A,l)(C,6)} is a subchronicle instance of *{{A,* 1)(B,3)(C,6)} but not of {(A,3)(C,6)}).

Subchronicle model: a model *C* is a subchronicle model of *C* (denoted by $C \sqsubseteq C$) *iff* from any instance of C, we can extract a chronicle instance of *C* (e.g., in Figure 2 the two right chronicle models are subchronicle models of the left one). This relation defines a partial order on chronicle models.

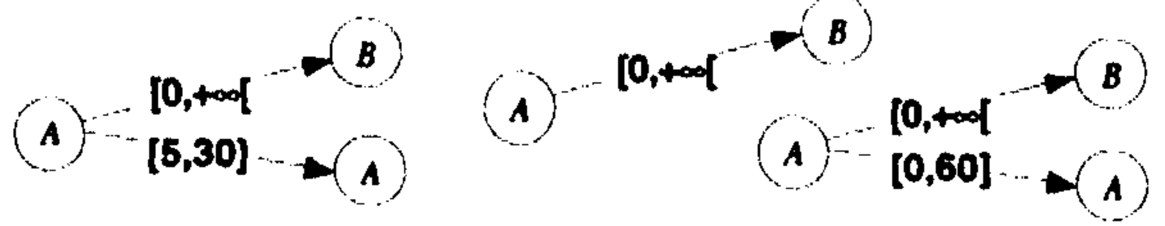

Figure 2: Chronicle and subchronicles models.

Superchronicle: *c* (resp. *C)* is a superchronicle instance (resp. model) of *c!* (resp. C), which is denoted by $c \sqsupseteq c'$ (resp. $C \sqsupseteq C'$), *iff c'* (resp. *C)* is a subchronicle instance (resp. model) of *c* (resp. *C).*

Theorem 1 *A chronicle model C* = (5, T) *is a subchronicle model of C* = (S',T') iff $S \subseteq S'$ *and* $\mathcal{T} \supseteq \mathcal{T}'$.

For brevity, hereafter *chronicle* stands for *chronicle model* and *instance* stands for *chronicle instance.*

[2]Of course, *ABB* is equivalent to *BAB* or *BBA.*

2.4 Instance Set and Frequency

Given an alarm log $\mathcal{L}$ and a recognition process, we denote $\mathcal{I}_C(\mathcal{L})$ the set of instances of C recognized in $\mathcal{L}$. Of course, this set depends strongly on the strategy implemented in the used recognition tool.

For our algorithms, we only need that the strategy guarantees the following property:

Property 2 $\forall c \in \mathcal{I}_C(\mathcal{L}),\ c' \in \mathcal{I}_C(\mathcal{L}),\ c \cap c' = \emptyset$.

This property means that an alarm occurrence can't belong to two instances of the *same* chronicle model.

For instance, with $\mathcal{L}$ given in Section 2.2, the instance set of the unconstrained chronicle *AB* could be:

- $\{\{(A,1)(B,3)\},\{(A,4)(B,8)\},\{(A,7)(B,10)\}\}$
- $\{\{(A,1)(B,12)\},\{(B,3)(A,7)\},\{(A,4)(B,10)\}\}$
- etc.

Frequency of chronicle: given an alarm log $\mathcal{L}$, the frequency $fq(C,\mathcal{L})$ of a chronicle C in $\mathcal{L}$ is the cardinal of $\mathcal{I}_C(\mathcal{L})$. For example, with $\mathcal{L}$ given in Section 2.2, the frequency of the unconstrained chronicle *AB* is 3.

Frequent chronicle: given a threshold fq_{min}, C is frequent in $\mathcal{L}$ if $fq(C,\mathcal{L}) \geq fq_{min}$. We suppose that the threshold fq_{min} is defined by the user.

As our algorithms work with only one alarm log at a time, we now use the simplified notations $\mathcal{I}_C$ and $fq(C)$ instead of respectively $\mathcal{I}_C(\mathcal{L})$ and $fq(C,\mathcal{L})$.

3 Discovering Frequent Chronicles

This section presents algorithms to discover all the sets of frequent chronicles of size *i* (denoted by Φ^i).

Lemma 3 *If a chronicle C is frequent in $\mathcal{L}$, all its subchronicles are also frequent in $\mathcal{L}$.*

A chronicle C^i won't be frequent if one of its subchronicles is infrequent. Otherwise, if all the subchronicles of C^i are frequent, C^i will be a *candidate* for Φ^i (i.e., C^i *may* be frequent). For example, with the unconstrained chronicle *ABC,* if one of its subchronicles *A, B, C, AB, AC, BC* is infrequent, *ABC* is also infrequent; otherwise, *ABC* may be frequent and is a candidate. In fact, if we know that *AB* is frequent, we also know that *A* and *B* are frequent. Thus we only need to check the frequency of *AB, AC, BC* (chronicles of size *i* - 1). We denote Φ^i_c the set of candidate chronicles for Φ^i ($\Phi^i_c \supseteq \Phi^i$).

So the intuitive solution to compute Φ^i_c is to generate all the possible chronicles C^i of size *i* from one of its frequent subchronicle of size *i* - 1 (if *AB* is infrequent, we do not need to generate AAB, ABB or ABC!) and then to check its frequency.

Based on these analyses, our main algorithm builds iteratively the exhaustive set of all frequent chronicles (see Figure 3). It starts by computing Φ^1, which corresponds to the set of all the frequent alarms in $\mathcal{L}$. At an iteration i, it first computes the set Φ^i_c from Φ^{i-1} (function *generateCandidate).* It then calculates the frequency of each candidate and keeps only the frequent ones in Φ^i. Its main loop ends at the iteration i where there is no frequent chronicle of size i ($i \leq |\mathcal{L}|$).

```
$\Phi^1 \leftarrow \{((A,t_A),\cdot) \mid A \in \mathcal{L}, fq(A) \geq fq_{min}\}$
while $\Phi^{i-1} \neq \emptyset$ do
        $\Phi^i_c \leftarrow generateCandidate(\Phi^{i-1})$
        $Calculate\ fq(C)\ for\ all\ the\ chronicles\ C\ of\ \Phi^i_c$
        $\Phi^i \leftarrow \{C \in \Phi^i_c \mid fq(C) \geq fq_{min}\}$
        $i \leftarrow i+1$
endwhile
```

Figure 3: Main algorithm

The following sections only give algorithms for the chronicle generation stage because we use CRS to calculate the frequency: chronicles are modeled into CRS, which receives $\mathcal{L}$ as alarm input stream and then the number of times of recognition performed by CRS for each chronicle gives us the frequency of the chronicle.

3.1 Candidate Generation

This section presents the algorithms to compute Φ^i_c from Φ^{i-1}.

There are two subtasks to build a candidate C^i: computing a set of *i* alarms occurring frequently together (i.e., a frequent unconstrained chronicle) and establishing the time constraints between these alarms. Therefore, in order to compute Φ^i_c, we first compute the set Ψ^i_c of the candidate unconstrained chronicles of size i that *may* be frequent. The set Ψ^i of frequent unconstrained chronicles of size i is then computed from Ψ^i_c. Eventually, we establish time constraints between alarms of Ψ^i to generate the final candidates of Φ^i_c (see Figure 4).

```
proc $generateCandidate(\Phi^{i-1})$
    $\Psi^i_c \leftarrow generateUnconstrainedCandidates(\Phi^{i-1})$
    $Calculate\ fq(C)\ for\ all\ the\ chronicles\ C\ of\ \Psi^i_c$
    $\Psi^i \leftarrow \{C \mid C \in \Psi^i_c\ and\ fq(C) \geq fq_{min}\}$
    $\Phi^i_c \leftarrow generateConstrainedCandidates(\Psi^i, \Phi^{i-1})$
return$(\Phi^i_c)$
```

Figure 4: Candidate generation

The frequency of unconstrained chronicles is easily computed by using the following lemma:

Lemma 4 *With an unconstrained chronicle C we have:* $fq(C) = \left\lfloor \min_{A_i \in C} \left(\frac{fq(A_i)}{N(A_i)} \right) \right\rfloor$, *where. N(Ai) is the number of alarms (of C) whose type is Ai.*

[3]Here, [r] stands for the greatest integer less than or equal to a real number r.

For example, with the alarm log given in Section 2.2, the frequency of the unconstrained *AAB* is: $\left[min\left\{\frac{fq(A)}{N(A)}=\frac{3}{2},\frac{fq(B)}{N(B)}=\frac{4}{1}\right\}\right]=1.$

Unconstrained Candidate Generation

First, Φ^{i-1} is used to construct the set Γ of the unconstrained chronicles corresponding to the chronicles of Φ^{i-1} (obviously, $\Gamma\subseteq\Psi^{i-1}$). We then compute the set Ψ_c^i from T. Lemma 3 enables us to suppress any chronicle including an infrequent subchronicle (see Figure 5).

```
proc generateUnconstrainedCandidates($\Phi^{i-1}$)
  $\Gamma = \{(\mathcal{S},\cdot) \mid \exists\mathcal{T}, (\mathcal{S},\mathcal{T}) \in \Phi^{i-1}\}$
  $\Omega$ = list of alarm types used in chronicles of $\Gamma$
  $\Psi_c^i \leftarrow \emptyset$
  for each $(\mathcal{S},\cdot)$ of $\Gamma$ do
    for each alarm type $A$ of $\Omega$ do
      $\mathcal{S}' = \mathcal{S}\cup\{(A,t_i)\}$
      if all subchronicles of $(\mathcal{S}',\cdot)$ are in $\Gamma$ then
        $\Psi_c^i \leftarrow \Psi_c^i \cup \{(\mathcal{S}',\cdot)\}$
return($\Psi_c^i$)
```

Figure 5: Unconstrained candidate generation

For example, with the alarm log given in Section 2.2, fq_{min} = 2 and *i* = 3, we have *T* = *{AB,AC,BC,BB}* and thus, Ω — *{A, B,C}*. By adding to *AB* each element of Ω, one by one, we generate the three candidates *AAB, ABB, ABC* for Ψ_c^3, but *AAB* is suppressed because one of its subchronicles, *A A*, is infrequent. We process similarly for *AC, BC, BB* and finally obtain Ψ_c^3= *{ABB, ABC, BBC)*.

With this exhaustive generation, one chronicle may be generated many times from those of *T* before checking its candidature for Ψ_c^i (e.g., *BBC* is generated twice, from *BB* and from *BC)*. To avoid this redundancy and also to reduce the number of generated chronicles, our optimized algorithm orders completely the alarm types of Ω (e.g. in lexical order) and only adds to a chronicle (5, •) of r an alarm whose type is greater than or equals to the greatest alarm type of *S* (see example in Figure 6). It can be proved that this optimization reduces in half the number of chronicles for which one needs checking the candidature for Ψ_c^i.

Establishing Time Constraints

Once Ψ^i is computed, the time constraints between the alarms of its chronicles are established by using Theorem 1. The idea is that the time constraints between two alarms in the chronicles of Φ_c^i are deduced from the time constraints between these alarms in the chronicles of Φ^{i-1} (see Figure 7).

Γ	exhaustive generation	optimized generation
AD	*AAB, ABB, ABC*	*ABB, ABC*
AC	*AAC,ABC,ACC*	*ACC*
BB	*ABB, BBB, BBC*	*BBB, BBC*
BC	*ABC, BBC, BCC*	*BCC*

Figure 6: Exhaustive vs. optimized generation for Ψ_c^3 (The bold chronicles are the candidates in Ψ_c^3).

```
proc generateConstrainedCandidates($\Psi^i$, $\Phi^{i-1}$)
  $\Phi_c^i \leftarrow \emptyset$
  for each $(\mathcal{S},.)$ of $\Psi^i$ do
    $\mathcal{T}$ is a constraint graph so that : $\forall(A,B)\in\mathcal{S}$,
    $K_{\mathcal{T}}(t_A \rightarrow t_B) = \bigcup_{(\mathcal{S}',\mathcal{T}')\in\Phi^{i-1}} K_{\mathcal{T}'}(t_A \rightarrow t_B)$
    if $\mathcal{T}$ is consistent then
      $\Phi_c^i \leftarrow \Phi_c^i \cup (\mathcal{S},\mathcal{T})$
return($\Phi_c^i$)
```

Figure 7: Computing Φ_c^i from Ψ^i and Φ^{i-1}

For example, suppose that in Φ^2 we have $K(t_A \rightarrow t_B)$ = [2,5], $K(t_B \rightarrow t_C)$ [1,1], $K(t_A \rightarrow t_C)$ [1,5], and the unconstrained chronicle *ABC* ∈ Ψ^3. The constraint graph $\mathcal{T}$ constructed from *{A,B,C}* with these constraints is consistent (see Figure 8), so $(\{(A,t_A),(B,t_B),(C,t_C)\},\mathcal{T})$ is a candidate in Φ_c^3.

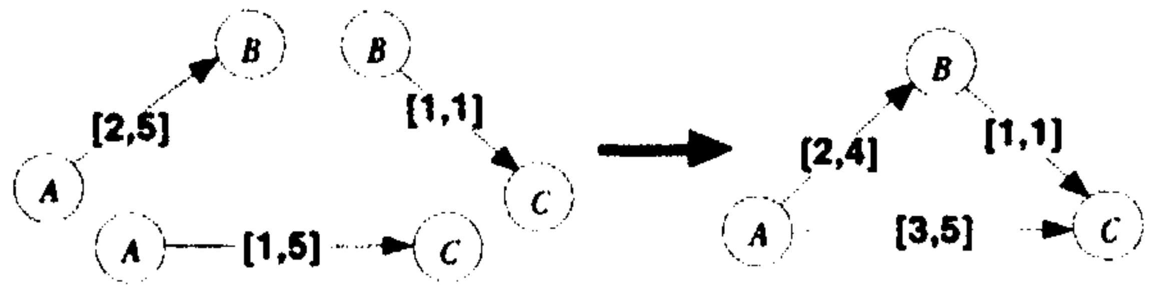

Figure 8: Establishing time constraints.

3.2 Discovering Chronicles of Size 2

The time constraints for the alarms are originally established at this stage and then are used to correlate alarms in the chronicles of size greater than 2. Using the same strategy as described above, our algorithm computes firstly Ψ^2, then establishes the time constraints between alarms of the chronicles of Ψ^2.

Based on the distinct instances of a frequent chronicle *AB* in the alarm log, we establish the time constraint between *A* and *B*. A constraint $[I^-,I^+]$ can be accepted as the time constraint between *A* and *B* if *all* the instances of $\mathcal{I}_{AB}$ respect $[I^-,I^+]$. But if *all* the instances of $\mathcal{I}_{AB}$ are used, some noises may be taken into account. In order to avoid noises, one should use an *instance threshold* it_{min} $(0<it_{min}\leq 1)$, i.e., only $\lceil it_{min}\times fq(AB)\rceil$ instances *of TAB* should be considered.

Among these *acceptable* time constraints, one should use some criteria to select the *good* ones for *A* and *B.* In order to find only tight constraints between alarms, we select only the constraints so that their *duration* and the *distance* between *A* and *B* are as small as possible since alarm effects are rapidly propagated in telecommunications network. More concretely:

- From the acceptable constraints, select $[I^-,I^+]$ so that (I^+-I^-) is the smallest (criterion of the tightest constraint).

- From the selected constraints, select $[I^-, I^+]$ so that $max\{|I^-|, |I^+|\}$ is the smallest (criterion of the shortest distance).

Searching good constraints (at most two) can be done using the algorithm *A**. These constraints can be considered as the *disjunctive* constraint between *A* and *B.* For example, with the instance threshold it_{min} = 1 and the alarm log $\mathcal{L}$ =(4,1)(B, *3)(A,* 4)(B, 5)(4, 7)(B, 8)(A 10), it is easy to find that [-2, -1] and [1,2] are two good time constraints between *A* and *B.*

In fact, CRS does not accept disjunctive constraints, therefore such ones will be unified to obtain the constraint for CRS. In the above example, the constraint $K(t_A \rightarrow t_B)$ will be [-2, 2] instead of -2, -1] ∪ [1,2] for CRS. Another possibility could be to define two chronicles *AB* in CRS, one with [-2,-1] and another with [1.2].

3.3 Discovering Frequent Chronicles with Chronicles Known by Experts

If we have some expertise, we can use it to improve the discovery process: all known chronicles of size *i* are added to Φ^i. Moreover, in order to determine whether the chronicle *C* is frequent, we use Theorem 1 to check if *C* is a subchronicle of one of the known chronicles. If it is, we already know that *C* is frequent. Otherwise, we have to calculate the frequency of C. Taking this remark into account, the number of chronicles to calculate the frequency may be much reduced.

3.4 Complexity

At an iteration i, the global complexity of our algorithms is the sum of the following complexities:

Unconstrained chronicle generation (Figure 5): for each unconstrained chronicle (S',•) of size i, checking if one of its subchronicles of size $i-1$ belongs to T can be done in time $O(i.\log|\Gamma|)$ with binary search; because $(S',\cdot)$ has *i* unconstrained subchronicles of size i — 1, checking the candidature of $(S',\cdot)$ f c Ψ^i_c a n be done in time $O(i^2.\log|\Gamma|)$. The number of unconstrained chronicles to check the candidature for Ψ^i_c is $|\Gamma|.|\Omega|$. In fact, $|\Gamma|$ can be overvalued to $|\Psi^{i-1}|$, so this complexity can be overvalued to $O(|\Psi^{i-1}|.|\Omega|.i^2.\log|\Psi^{i-1}|)$, which is a little better than one of [Mannila *et* al., 1995].

Unconstrained chronicles frequency calculation: it requires *Q(i)* time for a unconstrained candidate of size *i* (Lemma 4); so this complexity is $O(|\Psi^i_c|.i)$.

Establishing time constraints (Figure 7): for each unconstrained chronicle (5,) of Ψ^i, establishing the time constraint for one of its $\frac{i(i-1)}{2}$ pairs of alarms can be done in time $|\Phi^{i-1}|$; so, constructing a constraint graph corresponding to (5, •) requires $O(i^2.|\Phi^{i-1}|)$ time. In addition to this complexity, checking the consistency of a constraint graph of *i* nodes takes $O(i^3)$ time. Therefore this complexity is $O(|\Psi^i|.(i^2.|\Phi^{i-1}|+i^3))$.

Constrained chronicle frequency calculation : [Dousson, 1996] had shown that: for a chronicle of size i, the propagation when one of its alarms arrives requires $O(i^2)$ time; so we have a complexity of $O(i^3)$ for the chronicle; as the internal mechanism of CRS generates on average *i* instances for one recognized (and i^2 in the worst case), the average complexity of the frequency calculation is $O(i^4)$ (and $O(i^5)$ in the worst case). So this complexity is $O(|\Phi^i_c|.i^4)$ (and $O(|\Phi^i_c|.i^5)$ in the worst case).

4 Dependency Filtering

The number of frequent chronicles is always great. Thus, once these chronicles are discovered, one should filter them to find the relevant ones. What should be the criteria of relevance? Recall that our goal is to discover expertise for monitoring, so the discovered chronicles have to be able to identify phenomena produced during the functioning of the system. The question here is: does a chronicle signify itself a phenomenon or is it always included in a more complex one? For instance, is *AB* a phenomenon or does *AB always* come with other alarms (and so, the relevant chronicles may be *ABC* or *ABB)?*

We base the dependency relationship between a chronicle and one of its subchronicles on the alarms of the subchronicle that are not included in any instance of the superchronicle.

We define the *independent* instance set between two chronicles *C* and *C* as follows:

$$\mathcal{I}^\star_{C\sqsubseteq C'} = \begin{cases} \{c\in\mathcal{I}_C \mid (c \cap \bigcup_{c_i\in\mathcal{I}_{C_i}, C\sqsubset C_i\sqsubseteq C'} c_i) = \emptyset\} & \text{if } C\neq C' \\ \mathcal{I}_C & \text{if } C=C' \end{cases}$$

And so, the *independent* frequency $fq^\star(C\sqsubseteq C')$ is defined as the number of elements of $\mathcal{I}^\star_{C\sqsubseteq C'}$.

In term of dependency level, one can say that the chronicle *C depends completely* on all its subchronicles, i.e., the phenomenon corresponding to *C* included the phenomena corresponding to all its subchronicles. If there is *C* so that $fq^\star(C\sqsubseteq C')$ is high, one can say that the influence of C is not only showed in *C,* but also outside *C.* Therefore, if C' is a relevant chronicle, *C* is also relevant. On the other hand, if $fq^\star(C\sqsubseteq C')$ is low, one can say that outside C, the influence of *C* is not remarkable. In other words, *C* should only be considered as an excerpt of *C'.*

Theorem 5 $fq^\star(C\sqsubseteq C') \leq fq(C) - \sum_{C\sqsubset C_i\sqsubseteq C'} fq^\star(C_i\sqsubseteq C')$

Proof: This is due to the fact that for any instance c_i of C_i (C_i is superchronicle of C), there is at least one *dependent* instance *c* of *C* (i.e., $c\subset c_i$). □

Corollary 6 *The following recursive formula gives an* upper bound *of* $fq^\star(C\sqsubseteq C')$*:*

$$\widetilde{fq}(C\sqsubseteq C') = \begin{cases} fq(C) & \textit{if } C=C' \\ fq(C) - \sum_{C\sqsubset C_i\sqsubseteq C'} \widetilde{fq}(C_i\sqsubseteq C') & \textit{if } C\sqsubset C' \end{cases}$$

Proof: Recursively: suppose that the formula is true for any superchronicle C, Theorem 5 gives it for *C*. D

Let us suppose that the recognition process is contructed so that the instances of chronicles are recognized as soon as possible *(first-coming* instances). Namely, the process guarantees the following property (CRS does):

Property 7 *For any chronicle $\mathcal{C}$, its instance set $\mathcal{I}_\mathcal{C}$ is totally ordered by the relation $\prec$(before) between two instances of C, where $\prec$ is defined as follows:*

$$c = \{(A_i, t_i)\}, c' = \{(A_i, t'_i)\}, c \prec c' \text{ iff } \forall i, t_i < t'_i.$$

For example, with the alarm log given in Section 2.2, the instance set $\mathcal{I}_{AB}$ is the following: $\{\{(A,1)(B,3)\},\{(A,4)(B,8)\},\{(A,7)(B,10)\}\}$.

With our algorithms, we can prove the following theorem:

Theorem 8 ***For any frequent chronicle $\mathcal{C}'$ with distinct alarm types and $\mathcal{C}$ one of its subchronicles, if $\mathcal{I}_\mathcal{C}$ has Property 7, we have: $\widetilde{fq}(\mathcal{C} \sqsubseteq \mathcal{C}') = fq^*(\mathcal{C} \sqsubseteq \mathcal{C}')$.***

Thanks to Theorem 8, we can easily calculate the independent frequencies for chronicles with distinct alarm types. For the other kind of chronicles[4], we can calculate their independent frequencies by *exhaustive* algorithm (i.e., using the definition) and/or calculate the upper bounds for estimating purposes.

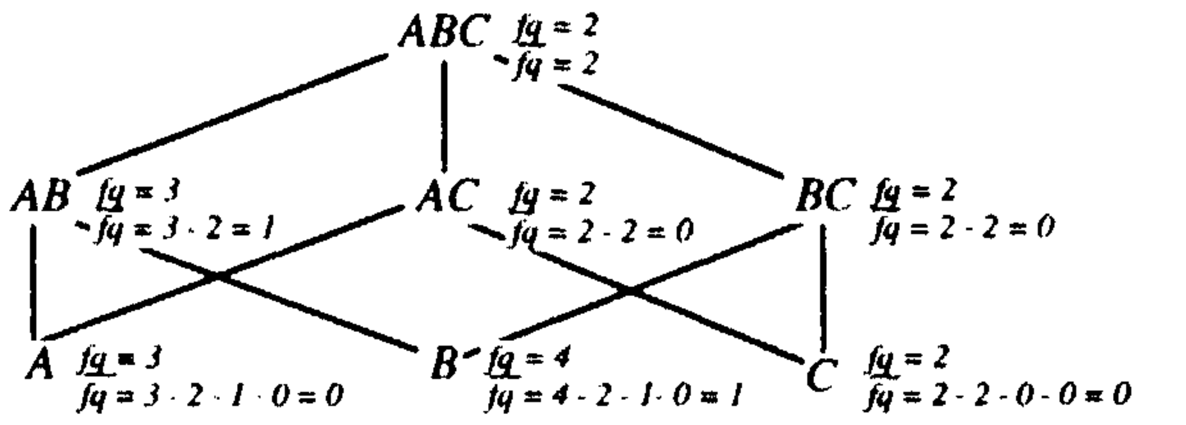

Figure 9: Calculation of $\widetilde{fq}(\mathcal{C} \sqsubseteq ABC)$ (with fq_min = 2).

Output of the Filtering Process

Based on the above analyses, the output of the filtering process is a graph *G* representing the dependency relationship between the frequent chronicles. There will be a link from C to C if $\mathcal{C} \sqsubseteq \mathcal{C}'$ and $fq^*(\mathcal{C} \sqsubseteq \mathcal{C}')$ = 0. [5]

A chronicle *C* is considered as *independent* if its independent frequencies against *all* its independent superchronicles greater than zero, i.e., *C* has no links with its independent superchronicles in *G.* All the chronicles that have no superchronicle are independent.

Figure 10 shows the dependency graph, which is computed from the alarm log $\mathcal{L}$ given in Section 2.2 with fq_{min} = 2. We exhibit here four independent chronicles ABB, *AB, ABC* and *BBC* (to simplify the figure, the corresponding constraint graphs are not shown).

[4] Fortunately, first experimental results show that this kind of chronicles is relatively rare.

[5]One can define a dependency threshold for filtering.

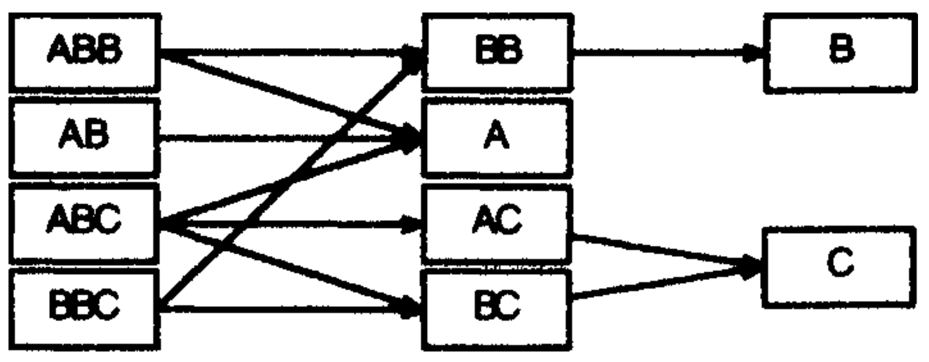

Figure 10: Dependency graph of the frequent chronicles.

5 Experiment and Results

The first experiment of our approach was with the data from the French packet switching telecommunications network. The input data was a log of 2900 alarms of 36 different types and corresponded to a duration of 20 hours. One run of our algorithms (i.e., for one given value of fq_{min}) took about 2 minutes, and many runs exhibited a dozen of independent relevant chronicles (due to the size of the input, frequent phenomena are relatively rare). Since the network is well known, this was more a validation experiment than an actual knowledge acquisition. In spite of that, one of the discovered chronicles was unknown but relevant to experts. Moreover, it was a non-trivial chronicle in the sense that it was out of usual monitoring knowledge: this chronicle showed the influence of a non-telecom unit - a secondary power supply failure - with an alarm indicating the too high temperature on an equipment (the reason is that the air conditioning system was out of order since it was plugged on the secondary power supply).

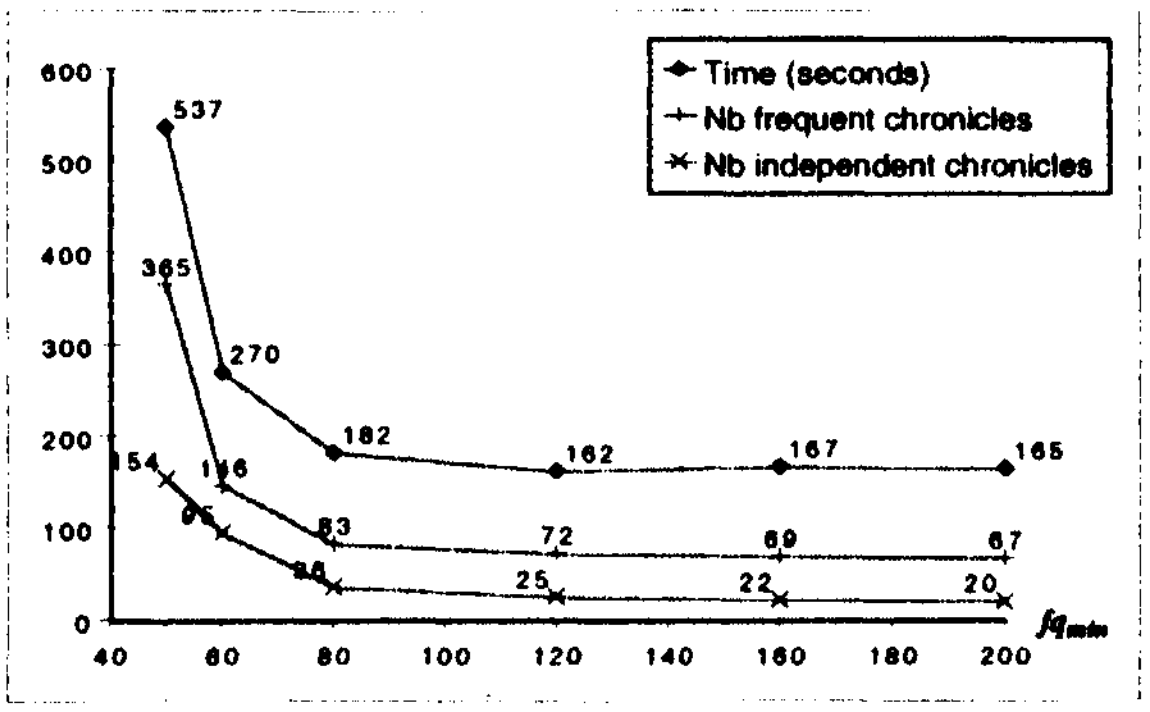

Figure 11: Experiment with ATM network data.

The second application is related to the first experimental national ATM (Asynchronous Transfer Mode) network. As this network is more recent, the challenge for our algorithms is to help efficiently experts with monitoring knowledge acquisition. The amount of data is more significant since we have a one month log with about forty thousand alarms dispatched through about 3800 different types. For this application, an alarm type consists of the actual alarm type and its localisation: for instance, if telecommunications equipment are able to emit a *LOS* (Loss of Signal), we define a *LOS-Lyon* for the Lyon switch, a *LOSJParis* for the Paris switch and soon . Figure 11 shows some experimental results. Moreover, we can notice that we have very few chronicles containing twice the same alarm type or more; hence, Theorem 8 (Section 4) gives us the exact independent frequency.

At this stage, we already know that some of them are relevant and some others must be filtered during on-line monitoring. Figure 1 (at the beginning of the paper) shows such a discovered chronicle: the first group *(LOF_{clear}, LOS_{clear}, LinkUp)* corresponds to the connection re-establishment and the second group *(LOF_{active}, LOSactive, Link Down)* indicates a connection breakdown; this chronicle signifies an unstable connection link, which fails in the 10 seconds following the re-establishment; this is a particular chronicle of the network and it is unpredictable according to the rules of recommendations of telecommunications management since it is a recurrent malfunctioning phenomenon. However, to improve the results, we need more investigations with telecommunications monitoring experts.

6 Conclusion

We present a chronicle discovery process that is very helpful for experts to discover monitoring knowledge from alarm logs. It is based on the key idea that a chronicle is frequent only if its subchronicles are frequent (like the work of [Mannila *et* a/., 1995] but the chronicle formalism is more expressive than their episode-rule structure). Our process is also able to take into account and to establish numerical time constraints between alarms since this information could be discrepant for the fault detection.

The use of the same chronicle recognition system and the same chronicle model guarantees that semantics of recognition remains the same during the off-line (knowledge acquisition) and on-line processing (monitoring).

We also propose a method for sorting the discovered chronicles and exhibiting the most relevant ones. The main advantage is to focus the attention of an expert on few (about one third) discovered chronicles to understand and to be categorized for the monitoring process.

The first experiments show that our algorithms are able to deal with an amount of data compatible with the requirement of a discovery process and the preliminary results are quite promising since many discovered chronicles could be explained by the corresponding ITU[7] recommendations. We will interact with equipment experts to validate more chronicles but we already know that this is a good way to acquire knowledge for monitoring a telecommunications network (like ATM). We noticed that the approach also exhibits some non-trivial scenarios, which are very rarely modelled by experts.

[6]This is a limitation of our algorithms: we only discover chronicles that correspond to the phenomena repeating on the same equipment. In future work, we will introduce variables in chronicles to capture similar phenomena occuring on different places in the network.

[7]International Telecommunication Union

Future work will focus on addition of variables in discovered chronicles and on introduction of some knowledge in the telecommunications domain to ease the chronicle generation stage. Experiments with the ATM network will also be pursued and another test on a SDH (Synchronous Digital Hierarchy) network is planned.

7 Acknowledgements

Thang Vu Duong is currently a Ph.D. student at the CNET (R&D centre of France Telecom), and is grateful to his advisor, Malik Ghallab (LAAS/CNRS, Toulouse), for helpful suggestions.

The authors would like to thank Thierry Vidal (ENIT, Tarbes), the GASPAR project team for useful discussions and especially, Philippe Dague (LIPN/CNRS, Paris) for valuable comments on this paper.

References

[Bibas *et* al., 1995] S. Bibas, M.O. Cordier, P. Dague, F. Levy, and L. Roze. Scenario generation for telecommunication network supervision. *Workshop on AI in Distributed Information Networks,* August 1995. Montreal, Quebec, Canada.

[Dechter *et* a/., 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence, Special Volume on Knowledge Representation,* 49(1-3):61-95, 1991. Elsevier Science B.V.

[Dousson *et* a/., 1993] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: Representation and algorithms. *Proc. of the 13th IJCAI,* pages 166-172, August 1993. Chambery, France.

[Dousson, 1996] C. Dousson. Alarm driven supervision for telecommunication networks : II- On-line chronicle recognition. In *Annals of Telecommunications,* 9/10:51, pages 501-508. CNET, France Telecom, October 1996.

[Jakobson and Weissman, 1995] G. Jakobson and M. Weissman. Real-time telecommunication network management: extending event correlation with temporal constraints. *Proc. 4th ISNIM,* pages 290-301, May 1995.

[Laborie and Krivine, 1997] P. Laborie and J.P. Krivine. Automatic generation of chronicles and its application to alarm processing in power distribution systems. *8th international workshop of diagnosis (DX'97),* September 1997. Mont Saint-Michel, France.

[Mannila *et al.,* 1995] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. *Proc. I^{Ht} KDD,* pages 210-215, August 1995. Montreal, Quebec:, Canada.

[Moller *et al,* 1995] M. Moller, S. Tretter, and B. Fink. Intelligent filtering in network-management systems. *Proc.* 4"^{1} *ISNIM,* pages 304-315, May 1995.

[Nygate, 1995] Y.A. Nygate. Event correlation using rule and object based techniques. *Proc. 4th ISNIM,* pages 279-289, May 1995.