Learning chronicles signing multiple scenario instances
Audine Subias, Louise Travé-Massuyès, Euriell Le Corronc

To cite this version:
Audine Subias, Louise Travé-Massuyès, Euriell Le Corronc. Learning chronicles signing multiple scenario instances. 25th International Workshop on Principles of Diagnosis - DX’14, Sep 2014, Graz, Austria. hal-01162866

HAL Id: hal-01162866
https://hal.science/hal-01162866
Submitted on 11 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Learning chronicles signing multiple scenario instances

Audine Subias\textsuperscript{1,3} and Louise Travé-Massuyès\textsuperscript{1,2} and Euriell Le Corronc\textsuperscript{1,4}
\textsuperscript{1}CNRS, LAAS, 7, avenue du Colonel Roche, F-31400 Toulouse, France
\textsuperscript{2}Univ de Toulouse, LAAS, F-31400 Toulouse, France
\textsuperscript{3}Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
\textsuperscript{4}Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France
e-mail: {subias, louise, elecorronc}@laas.fr

Abstract

Chronicle recognition is an efficient and robust method for fault diagnosis. The knowledge about the underlying system is gathered in a set of chronicles, then the occurrence of a fault is diagnosed by analyzing the flow of observations and matching this flow with a set of available chronicles. The chronicle approach is very efficient as it relies on the direct association of the symptom, which is in this case a complex temporal pattern, to a situation. Another advantage comes from the efficiency of recognition engines which make chronicles suitable for one-line operation. However, there is a real bottleneck for obtaining the chronicles. In this paper, we consider the problem of learning the chronicles. Because a given situation often results in several admissible event sequences, our contribution targets an extension to multiple event sequences of a chronicle discovery algorithm tailored for one single event sequence. The concepts and algorithms are illustrated with representative and easy to understand examples.

1 Introduction

Chronicles are temporal patterns well suited to capture the behavior of dynamic processes at an abstract level based on events. They are among the formalisms that can be used to model timed discrete event systems. Chronicles may represent the signatures of specific situations, and are hence very efficient for diagnosis. They may also be associated to decision rules specifying which actions must be undertaken when reconfiguration is needed.

The chronicle approach has been developed and used in a large spectrum of diagnosis applications [1]: in the medical field for ECG interpretation and cardiac arrhythmia detection [2], in intrusion detection systems [3], in telecommunication networks [4]. More recently, chronicles have been used in the context of web services [5, 6]. Chronicles are also applied on activity recognition in the setting of unmanned aircraft Systems and unmanned aerial vehicles operating over road and traffic networks [7].

Among the challenges raised by the chronicle approach is the problem of acquiring the chronicles. On one hand, model based chronicle generation approaches have been developed. For instance, in [8] the patterns are built from Petri net models of the situation to recognize. Nevertheless, most of the works addressing this problem are data driven. They rely on analyzing logs and extract the significant patterns by temporal data mining techniques [9].

The objective of temporal data mining techniques is to discover all patterns of interest in the input data, by means of an unsupervised approach. There are several ways to define the relevance of a pattern. Among them, the frequency criterium is widely used [10, 11]. One can distinguish two main frameworks: sequential patterns [12] and frequent episodes [13].

- The sequential pattern framework is based on the discovery in a collection of sequences of all possible time ordered sets of event (i.e. sequence of events) with sufficient number of occurrences w.r.t a user-defined threshold. The number of occurrences of a set of events is defined as the number of times the set of events can be observed in the collection. Further, a sequence of events is said to be maximal if it involves the highest possible number of events. Sequential pattern discovery relies on the systematic search of maximal sequences that have a number of occurrences at least equal to a user-defined threshold. Many methods for unearth sequential patterns are designed along the lines of the Apriori algorithm [12].

- Frequent episode framework uses a single (long) sequence and considers the discovery of temporal patterns, called episodes, that occur with sufficient frequency in the sequence. An episode is a partially ordered set of event types. Similarly to the case of sequential patterns, the notion of frequent episode and sub-episode are defined. In [13] episode discovery focuses on two types of episodes: serial episodes when the order between the event types is total and parallel episodes when there is only partial order between the events types.

Chronicles are designed to afford for total order and partial order temporal patterns but also patterns combining the two types. Moreover, chronicles consider temporal constraints between event type occurrences.

One of the main difficulties of chronicle discovery is to guarantee robustness to variations. The chronicle discovery approach proposed in this paper aims at discovering frequent chronicles common to multiple sequences representing variations of a unique situation, that is to say chronicles that are frequent in each sequence of a collection. This is motivated by the fact that the event sequences arising from the same situation generally present variants that must be accounted for.
The temporal sequence *typology* is the set of event types $E' \subseteq E$ that occur in $S$. Moreover, let us point out that in such temporal sequences, the index $i$ refers to the event type $e_i$ whereas the index $j$ refers to the chronology of the event dates.

*Example:* Let us consider the following temporal sequence $S = \{(e_1, t_1), (e_2, t_2), (e_1, t_3)\}$ where $l = 3$. The event date of $(e_1, t_1)_2$ is lower than the event date of $(e_2, t_2)_2$, itself lower than the event date of $(e_1, t_3)_3$.

To represent specific domain evolutions, we consider that event dates may be constrained. Consider two events $(e_i, t_i)$ and $(e_j, t_j)$. We define $I$ as the time interval expressed as the pair $I_{ij} = [t^-, t^+]$ corresponding to the lower and upper bounds on the temporal distance between the two event dates $t_i$ and $t_j$, $t^- \leq t^+$. We denote by $\tau_{ij}$ the temporal constraints defined by $\tau_{ij} = t_j - t_i \in [t^-, t^+]$. If the event dates $t_i$ and $t_j$ satisfy the temporal constraint $\tau_{ij}$, we write $e_i[t^-, t^+]e_j$ and say that the events are temporally constrained by $\tau_{ij}$. $I_{ij} = [t^-, t^+]$ is called the *support* of $\tau_{ij}$.

Several events may have the same event type and hence a pair $(e_i, e_j)$ may not be unically temporally constrained. Consequently, temporal constraints $\tau_{ij}$ and their corresponding supports $I_{ij}$ are indexed by $k$ as $\tau_{ij}^k$ and $I_{ij}^k$. For sake of simplicity, the index $k$ is avoided when not necessary.

**Definition 3 (Chronicle).** A chronicle is a triplet $C = (\mathcal{E}, \mathcal{T}, \mathcal{G})$ such that $\mathcal{E} \subseteq E$ with $\forall e_i \in \mathcal{E}$, $\forall e_i \in \mathcal{G}$, $e_{i+1}$, $\mathcal{T} = \{\tau_{ij}\}_{i < j \leq |\mathcal{E}|}$, and $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a directed graph where the nodes $\mathcal{V}$ represent event types of $\mathcal{E}$ and the arcs $\mathcal{A}$ represent the constraints between the event dates. $\mathcal{E}$ is called the *typology* of the chronicle, $\mathcal{T}$ is the set of temporal constraints of the chronicle, and $\mathcal{G}$ is the *precedence graph*.

*Example:* The chronicle concept can be illustrated by the chronicle $C$ defined by $\mathcal{E} = \{e_1, e_2, e_3\}$, $\mathcal{T} = \{\tau_{12}, \tau_{12}, \tau_{23}\}$, and $\mathcal{G}$ given in Figure 1. Moreover, we have $I_{12} = [1, 3]$, $I_{12} = [2, 10]$ and $I_{23} = [4, 6]$.

![Figure 1: Chronicle precedence graph $G$](image)

A chronicle $C$ represents an evolution pattern involving a subset of event types $\mathcal{E}$, a set of temporal constraints $\mathcal{T}$ linking event dates, and a directed graph $\mathcal{G}$ linking event occurrences. Chronicles define temporal patterns similar to temporal constraint networks [15], where the temporal order of events is quantified with numerical bounds and reflects the represented piece of temporal evolution.

The *episodes* of $[13]$ are a particular type of chronicle. A parallel episode is a collection of event types that occur in a given partial order whereas the event types of a serial episode are totally ordered. In the case of episodes, temporal constraints do not specify numerical temporal bounds but are just precedence constraints of the form $t_j - t_i \in [0, +\infty]$ or $t_j - t_i \in ]-\infty, 0]$. An episode can hence be viewed
as a degenerated chronicle defined by the pair \((E, G)\). Conversely, chronicles are parallel episodes with additional temporal information.

The occurrences of a given chronicle \(C\) in a temporal sequence \(S\) are denoted by subsequences called chronicle instances.

**Definition 4** (Chronicle instance). An instance of \(C = (E, T, G)\) in a temporal sequence \(S\) is a subset of event types \(E'\) of \(S\) such that \(E'\) is isomorphic to \(E\), in other words \(|E'| = |E|\), and the event types of \(E'\) satisfy all temporal constraints \(T\) of the chronicle \(C\) according to the graph \(G\).

**Definition 5** (Frequency of a chronicle). The frequency of a chronicle \(C\) in a temporal sequence \(S\), noted \(f(C|S)\), is the number of instances of \(C\) in \(S\).

In the literature, it is hardly the case that all the instances of a given chronicle are returned by a chronicle recognition engine. Additional constraints are generally used to form a recognition criterion \(\gamma\). For example, [13] returns the shortest instances in the sense of the instance duration in the sequence. [10] returns all the recognized at the earliest disjoint instances, i.e., all the instances that do not overlap in the sequence and that occur earliest in the sequence. The frequency of a chronicle \(C\) in a temporal sequence \(S\) can be defined according to the recognition criterion \(\gamma\) and it is then noted \(f_\gamma(C|S)\).

Given a set of event types \(E\), the space of possible chronicles can be structured by a generality relation.

**Definition 6** (Generality relation among chronicles. A chronicle \(C = (E, T, G)\) is more general than a chronicle \(C' = (E', T', G')\), denoted \(C \sqsubseteq C'\), if \(E \subseteq E'\) or \(\forall \tau_{ij} \in T, \tau_{ij} \supseteq \tau_{ij}'\) or \(G'\) is a subgraph of the transitive closure of \(G\). Equivalently, \(C'\) is said stricter than \(C\).

**Definition 7** (Monotonicity). A frequency \(f_\gamma\) is said to be monotonic for the relation \(\sqsubseteq\) if and only if \(C \sqsubseteq C'\) implies that \(f_\gamma(C|S) \geq f_\gamma(C'|S)\) for any temporal sequence \(S\) of events.

**Definition 8** (Minimal and maximal chronicles of a set). Given a set of chronicles \(C\), the subset of minimal and maximal chronicles of \(C\) are defined by:

\[
\text{Min}(C) = \{ C \in C | \forall C' \in C, C \sqsubseteq C' \Rightarrow C = C' \},
\]

\[
\text{Max}(C) = \{ C \in C | \forall C' \in C, C \sqsubseteq C' \Rightarrow C = C' \}.
\]
\((e_1, e_2)\) for each sequence are given by \(f^1_{12} = 3, f^2_{12} = 2, f^3_{12} = 3\), hence \(f_{max} = 2\).

**Figure 2:** Example of multiple sequences

The roots of the trees for the pair \((e_i, e_j)\) must hence be such that the number of occurrences \(\{(e_i, t_i), (e_j, t_j)\}\) in all temporal sequences \(S\) is equal to \(f_{max}\). The following explains how to obtain these roots.

Two sets of supports are considered for each sequence \(S_k\): the set of minimal supports \(\bar{I}^k_{ij}\) and the set of maximal supports \(\bar{M}^k_{ij}\) that guaranty exactly \(f_{max}\) occurrences of the pair \((e_i, e_j)\) in \(S_k\). Whereas minimal supports are used in the algorithm of [11], maximal supports are a new concept required by our method to deal with multiple sequences. These sets are defined as follows:

\[
\bar{I}^k_{ij} = \{f^k_{ij}, \cdots, f^n_{ij}\} \quad f^k_{ij} = f_{max}
\]

and \(\forall [t^-, t^+] \subseteq [t^-, t^+] \quad f^k_{ij} < f_{max}\).

\[
\bar{M}^k_{ij} = \{\bar{T}^k_{ij}, \cdots, \bar{T}^n_{ij}\} \quad f^k_{ij} = f_{max}
\]

and \(\forall [t^-, t^+] \supseteq [t^-, t^+] \quad f^k_{ij} > f_{max}\).

**Example:** On the example of Figure 2, the minimal and maximal supports that guaranty exactly \(f_{max} = 2\) for each sequence are \(\bar{I}^1_{12} = \{[-2, 1], [1, 2]\}\) and \(\bar{M}^1_{12} = \{[-1, 1], [-1, +\infty]\}\), \(\bar{I}^2_{12} = \{[1, 1]\}\) and \(\bar{M}^2_{12} = \{[-\infty, +\infty]\}\), \(\bar{I}^3_{12} = \{[-1, 1], [1, 3]\}\) and \(\bar{M}^3_{12} = \{[-\infty, 2], [1, +\infty]\}\).

Then, the minimal and maximal supports obtained for each \(S_k\) are combined to obtain all the possibilities for the whole set of sequences \(S\). Let us denote by \(\bar{I}_{ij}^{comb}\) and \(\bar{M}_{ij}^{comb}\) the set of minimal and maximal support combinations, respectively:

\[
\bar{I}_{ij}^{comb} = \{L_{ij}^{comb}, \cdots, L_{ij}^{n}\} \quad L_{ij}^{k} \in \bar{I}^{k}_{ij}, k = 1, \cdots, n
\]

\[
\bar{M}_{ij}^{comb} = \{\bar{T}_{ij}^{comb}, \cdots, \bar{T}^{n}_{ij}\} \quad \bar{T}_{ij}^{k} \in \bar{M}^{k}_{ij}, k = 1, \cdots, n
\]

The union of the minimal supports of every combination of \(\bar{I}_{ij}^{comb}\) is a candidate for being a tree root for \((e_i, e_j)\). The set of tree root candidates for \((e_i, e_j)\) is given by:

\[
\bar{R}_{ij} = \{r^\alpha_{ij} = \bigcup_k \bar{I}^k_{ij} \in \bar{I}_{ij}^{comb}, k = 1, \cdots, n, \alpha = 1, \ldots, \text{card}(\bar{I}_{ij}^{comb})\}
\]

However, minimal supports are determined independently for each sequence. As a consequence, a candidate may violate the \(f_{max}\) occurrences rule in some of the sequences, in which case it is not valid.

The validity of a candidate can be assessed thanks to the maximal supports. Indeed, the intersection of the maximal supports of every combination of \(\bar{I}_{ij}^{comb}\) provides a **maximal common interval**, denoted \(\text{MCI}_{ij}\), that guaranties exactly \(f_{max}\) occurrences of the pair \((e_i, e_j)\) in all the sequences \(S_k \in S\):

\[
\text{MCI}_{ij} = \{\text{MCI}_{ij}^\beta = \bigcap_k T^k_{ij} \in T^k_{ij}, k = 1, \cdots, n, \beta = 1, \ldots, \text{card}(\bar{I}_{ij}^{comb})\}
\]

**Property 1.** A tree root candidate \(r^\alpha_{ij}\) of \(\bar{R}_{ij}\) is valid if \(\exists \beta\) such that \(r^\alpha_{ij} \subseteq \text{MCI}_{ij}^\beta\), where \(\text{MCI}_{ij}^\beta \in \text{MCI}_{ij}\).

A valid tree root candidate for \((e_i, e_j)\) is denoted \(R^\alpha_{ij}\) and the set of such roots is denoted \(R_{ij}\). The cardinal of \(R_{ij}\) is \(n_{ij}\).

**Example:** Let us build the first following combinations by taking the first elements of the minimal and maximal supports for each sequence: \{[-2, 1], [-1, 1], [-1, 1]\} \in \bar{I}_{12}^{comb} and \{[-\infty, 1], -\infty, +\infty[, 1, +\infty[\} \in \bar{M}_{12}^{comb}\). The intersection of the maximal supports provides a maximal common interval \(\text{MCI}_{12}^1\) that validates the union of the minimal supports \(r^1_{12}\), that is:

\[
r^1_{12} = [-2, 1] \subseteq \text{MCI}_{12}^1 = [-\infty, 1]
\]

Hence, a first valid tree root for pair \((e_1, e_2)\) is given by \(R^1_{12} = r^1_{12} = [-2, 1]\). However, about the second obtained combinations \{[-2, 1], [-1, 1], [1, 3]\} \in \bar{I}_{12}^{comb} and \{[-\infty, 1], -\infty, +\infty[, 1, +\infty[\} \in \bar{M}_{12}^{comb}\), the intersection of the maximal supports provides a maximal common interval \(\text{MCI}_{12}^2 = [1, 1]\) that do not validate the union of the minimal supports \(r^2_{12} = [-2, 3]\). Hence, \(r^2_{12}\) is not a good candidate to be a tree root, it authorizes \(3\) occurrences in sequences \(1\) and \(\beta\) so it violates the \(f_{max} = 2\) occurrences rule.

The same procedure is applied for \(f = f_{max} = 1\) and so on until \(f = 1\) to find the tree roots in case no candidate is valid for \(f + 1\) or to find the child nodes of the lower levels of the trees. The lowest level always corresponds to \(f = 1\) and is never empty because the considered pairs have been taken as pairs appearing in all the sequences. The trees for the different pairs \((e_i, e_j)\) may have a root corresponding to a different frequency. The root frequency for the trees of \((e_i, e_j)\) is denoted \(f^\text{root}_{ij}\).

**Example:** For the example of Figure 2, two valid tree roots are found for pair \((e_1, e_2)\): \(R^1_{12} = [-2, 1]\) and \(R^2_{12} = [-1, 2]\). The set of trees is illustrated in Figure 3 with \(f^\text{root}_{ij} = 2\) and \(n_{ij} = 2\).

### 3.2 Chronicle discovery algorithm

Once the constraint database \(D\) has been built to account for all the sequences in \(S\) as presented in section 3.1, the algorithm for discovering all minimal frequent chronicles for \(D\), given an input frequency threshold \(f_{th}\), is the same as the HCDA algorithm of [11] but the counting step. This
The algorithm is hence called HCDA-modified (HCDAM). The counting step is slightly different because we want HCDAM to provide all the frequent chronicles whose frequency is above the specified threshold. Like in [11], the candidate chronicles are called $\mathbb{D}$-chronicles.

The principle of HCDAM is to generate a set of candidate $\mathbb{D}$-chronicles from a chronicle that was proved to be frequent. The set Candidates is initiated with the set of root trees $\mathbb{D}^T$. The algorithm maintains two lists:

- the list $\text{Frequent}$ includes the candidate $\mathbb{D}$-chronicles that have been proved frequent, i.e. whose frequency is higher or equal to $f_{th}$, and strictest,
- the list $\text{NotFrequent}$ includes the chronicles that have been proved not frequent.

Instead of immediately counting a candidate $\mathbb{C}$, i.e. determining the minimal number of occurrences in the sequences of $\mathbb{S}$, the algorithm first makes use of the generality relation and monotonicity property to discard or accept the candidate without counting:

- if there exists a chronicle $\mathbb{C}'$ more general than $\mathbb{C}$ in $\text{NotFrequent}$, then $\mathbb{C}$ is not frequent as well,
- if there exists a chronicle $\mathbb{C}'$ stricter than $\mathbb{C}$ in $\text{Frequent}$, then $\mathbb{C}$ is also frequent.

If none of the two above situations apply, then $\mathbb{C}$ is counted, which is performed with CRS (Chronicle Recognition System) [14].

In our algorithm, the candidate is also counted in the second situation to determine its actual frequency, which is necessarily higher than $f_{th}$ and higher than the frequency of the stricter chronicle $\mathbb{C}'$. Obviously, only maximal chronicles are saved in $\text{NotFrequent}$ because this set is used to search for more general chronicles. Conversely, only minimal chronicles are saved in $\text{Frequent}$.

Example: Consider the following sequences of events.

<table>
<thead>
<tr>
<th>Sequence 1</th>
<th>Sequence 2</th>
<th>Sequence 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$: 1.049432</td>
<td>$e_2$: 1.606904</td>
<td>$e_3$: 3.410456</td>
</tr>
<tr>
<td>$e_2$: 1.4737624</td>
<td>$e_4$: 2.1878412</td>
<td>$e_5$: 18.8950274</td>
</tr>
<tr>
<td>$e_3$: 2.8735412</td>
<td>$e_5$: 15.1968734</td>
<td>$e_6$: 18.3950527</td>
</tr>
<tr>
<td>$e_4$: 2.8735412</td>
<td>$e_5$: 15.1968734</td>
<td>$e_6$: 18.3950527</td>
</tr>
</tbody>
</table>

The algorithm HCDA-modified has been used to learn the two following chronicles:

- $\mathbb{C}_1 = (E_1, T_1, \mathcal{G}_1)$, with $E_1 = \{e_2, e_3, e_4, e_5\}$, $T_1 = \{\tau_{52}, \tau_{54}, \tau_{24}, \tau_{53}, \tau_{23}\}$ and $\mathcal{G}_1$ given in Figure 4.

- $\mathbb{C}_2 = (E_2, T_2, \mathcal{G}_2)$, with $E_2 = \{e_4, e_5\}$, $T_2 = \{\tau_{54}\}$ and $\mathcal{G}_2$ given Figure 5.

Figure 4: Graph $\mathcal{G}_1$ of chronicle $\mathbb{C}_1$.

Figure 5: Graph $\mathcal{G}_2$ of chronicle $\mathbb{C}_2$.

The second chronicle is a chronicle of frequency 2, which means that it must be recognized twice to sign the situation captured by the three sequences of the example. The first chronicle is the stricter chronicle. It involves the larger set of event types with the tightest temporal constraints. It is of frequency 1.

4 Conclusion

This paper deals with the problem of discovering temporal patterns in the form of chronicles that are common to a set of temporal sequences issued from the same situation. The obtained chronicles then sign the situation and can be used for situation assessment or diagnosis purposes. The paper builds on work by [11] and extends the proposed algorithm that targets learning the frequent chronicles for one single sequence to multiple temporal sequences that represent variants of a unique situation. This requires to deeply revise the algorithm to generate the constraint database representing the temporal sequences. The revised algorithm is illustrated by a simple example which helps understand the different steps of the method.

Future research includes theoretical as well as applied work. The complexity of the modified algorithm, in particular the way to build the constraint database, should be carefully analyzed and ways to improve efficiency should be studied. The theoretical complexity of HCDAM is clearly the same as the one of [11] but dealing with multiple sequences reduces the number of constraint graphs in the database as well as the number of possible temporal constraints between each pair of events, resulting in increased tractability. Actually, we believe that improving the algorithm’s complexity is rather to be expected from a more efficient way to generate the constraint database and this is one of our goal for the near future.

On the other hand, it is planned to use this work for a real prognostic problem, applying the algorithm HCDAM to the observable signals of a pressure regulation valve of a modern aircraft in different wear situations.

References


