First-Order Logic Based Formalism for Temporal Data Mining*

Paul Cotofrei[1] and Kilian Stoffel[2]

[1] University of Neuchâtel, Pierre-à-Mazel, 7, 2000, Neuchâtel, Switzerland
paul.cotofrei@unine.ch
[2] University of Neuchâtel, Pierre-à-Mazel, 7, 2000, Neuchâtel, Switzerland
kilian.stoffel@unine.ch

Summary. In this article we define a formalism for a methodology that has as purpose the discovery of knowledge, represented in the form of general Horn clauses, inferred from databases with a temporal dimension. To obtain what we called *temporal rules*, a discretisation phase that extracts *events* from raw data is applied first, followed by an induction phase, which constructs classification trees from these events. The theoretical framework we proposed, based on first-order temporal logic, permits us to define the main notions (event, temporal rule, constraint) in a formal way. The concept of *consistent linear time structure* allows us to introduce the notions of *general interpretation* and of *confidence*. These notions open the possibility to use statistical approaches in the design of algorithms for inferring higher order temporal rules, denoted temporal meta-rules.

1 Introduction

Data mining is the process of discovering interesting knowledge, such as patterns, associations, changes, anomalies and significant structures, in large amounts of data stored in databases, data warehouses, or other data repositories. Due to the wide availability of huge amounts of data in digital form, and the need for turning such data into useful information, data mining has attracted a great deal of attention in information industry in recent years.

In many applications, the data of interest comprise multiple sequences that evolve over time. Examples include financial market data, currency exchange rates, network traffic data, signals from biomedical sources, etc. Although traditional time series techniques can sometimes produce accurate results, few can provide easy understandable results. However, a drastically increasing number of users with a limited statistical background would like to use these tools. In the same time, we have a number of tools developed by researchers in

*Supported by the Swiss National Science Foundation (grant N ° 2100-063 730).

the field of artificial intelligence, which produce understandable rules. However, they have to use ad-hoc, domain-specific techniques for transforming the time series to a "learner-friendly" representation. These techniques fail to take into account both the special problems and special heuristics applicable to temporal data and therefore often results in unreadable concept description.

As a possible solution to overcome these problems, we proposed [9] a methodology that integrates techniques developed both in the field of machine learning and in the field of statistics. The machine learning approach is used to extract symbolic knowledge and the statistical approach is used to perform numerical analysis of the raw data. The overall goal consists in developing a series of methods able to extract/generate temporal rules, having the following characteristics:

- Contain explicitly a temporal (or at least a sequential) dimension.
- Capture the correlation between time series.
- Predict/forecast values/shapes/behavior of sequences (denoted events).

1.1 State of Art

The domain of temporal data mining focuses on the discovery of causal relationships among events that may be ordered in time and may be causally related. The contributions in this domain encompass the discovery of temporal rule, of sequences and of patterns. However, in many respects this is just a terminological heterogeneity among researchers that are, nevertheless, addressing the same problem, albeit from different starting points and domains.

The main tasks concerning the information extraction from time series database and on which the researchers concentrated their efforts over the last years may be divided in several directions. *Similarity/Pattern Querying* concerns the measure of similarity between two sequences or sub-sequences respectively. Different methods were developed, such as window stitching [1] or dynamic time warping based matching [24, 23]. *Clustering/Classification* concentrates on optimal algorithms for clustering/classifying sub-sequences of time series into groups/classes of similar sub-sequences. Different techniques were proposed: Hidden Markov Models (HMM) [25], Dynamic Bayes Networks (DBNs) [13], Recurrent Neural Networks [14], supervised classification using piecewise polynomial modelling [30] or agglomerative clustering based on enhancing the time series with a line segment representation [22]. *Pattern finding/Prediction* methods concern the search for periodicity patterns (fully or partially periodic) in time series databases. For full periodicity search there is a rich collection of statistic methods, like FFT [26]. For partial periodicity search, different algorithms were developed, which explore properties related to partial periodicity such as the max-subpattern-hit-set property [16] or point-wise periodicity [15]. *Temporal Rules' extraction* approach concentrated to the extraction of explicit rules from time series, like temporal association rules [6] or cyclic association rules [31]. Adaptive methods for finding rules

whose conditions refer to patterns in time series were described in [11, 37, 17] and a general methodology for classification and extraction of temporal rules was proposed in [9].

Although there is a rich bibliography concerning formalism for temporal databases, there are very few articles on this topic for temporal data mining. In [2, 5, 29] general frameworks for temporal mining are proposed, but usually the researches on causal and temporal rules are more concentrated on the methodological/algorithmic aspect, and less on the theoretical aspect. In this article, we extend our methodology with an innovative formalism based on first-order temporal logic, which permits an abstract view on temporal rules. The formalism allows also the application of an inference phase in which higher order temporal rules (called temporal meta-rules) are inferred from local temporal rules, the lasts being extracted from different sequences of data. Using this strategy, known in the literature as higher order mining [36], we can guarantee the scalability (the capacity to handle huge databases) of our methodological approach, by applying statistical and machine learning tools.

It is important to mention that there are two distinct approaches concerning the time structure in a framework for temporal data mining. The first conceives the time as an ordered sequence of points and it is usually employed in temporal database applications. The second is based on intervals and it is predominant in AI applications [3, 8, 35, 19]. The difference induced at the temporal logic level, by the two approaches, is expressed in the set of temporal predicates: they are unary in the first case and binary for the second.

The rest of the paper is structured as follows. In the next section, the main steps of the methodology (the discretisation phase and the inference phase) are presented. The Section 3 contains an extensively description of the first-order temporal logic formalism (definitions of the main terms – *event, temporal rules, confidence* – and concepts – *consistent linear time structure, general interpretation*). The Section 4 reviews the methodology for temporal rules extraction, in the frame of the proposed formalism. The notion of temporal meta-rules and the algorithms for inferring such high order rules are described in Section 5. Finally, the last section summarizes our work and emphasize what we consider an important and still open problem of our formalism.

2 The Methodology

The approaches concerning the information extraction from time series, described above, have mainly two shortcomings, which we tried to overcome.

The first problem is the type of knowledge inferred by the systems, which most often is difficult to be understood by a human user. In a wide range of applications, (e.g. almost all decision making processes) it is unacceptable to produce rules that are not understandable for an end user. Therefore, we decided to develop inference methods that produce knowledge represented

in the form of general Horn clauses, which are at least comprehensible for a moderately sophisticated user. In the fourth approach, (*Temporal Rules' extraction*), a similar representation is used. However, the rules inferred by these systems have a more restricted form than the rules we propose.

The second problem consists in the number of time series investigated during the inference process. Almost all methods mentioned above are based on one-dimensional data, i.e. they are restricted to one time series. The methods we propose are able to handle multi-dimensional data.

Two of the most important scientific communities which brought relevant contributions to data analysis (the statisticians and database researchers) chose two different ways: statisticians concentrated on the continuous aspect of the data and the large majority of statistical models are continuous models, whereas the database community concentrated much more on the discrete aspects, and in consequence, on discrete models. For our methodology, we adopt a mixture of these two approaches, which represents a better description of the reality of data and which generally allows us to benefit from the advantages of both approaches.

The two main steps of the methodology for temporal rules extraction are structured in the following way:

1. Transforming sequential raw data into sequences of events: Roughly speaking, an event can be seen as a labelled sequence of points extracted from the raw data and characterized by a finite set of predefined features. The features describing the different events are extracted using statistical methods.
2. Inferring temporal rules: We apply a first induction process, using sets of events as training sets, to obtain several classification trees. Local temporal rules are then extracted from these classification trees and a final inference process will generate the set of temporal meta-rules.

2.1 Phase One

The procedure that creates a database of events from the initial raw data can be divided into two steps: time series discretisation, which extracts the discrete aspect, and global feature calculation, which captures the continuous aspect.

Time series discretisation. During this step, the sequence of raw data is "translated" into a sequence of discrete symbols. By an abuse of language, an event means a subsequence having a particular shape. In the literature, different methods were proposed for the problem of discretizing times series using a finite alphabet (window's clustering method [11], ideal prototype template [22], scale-space filtering [18]). In window's clustering method, all contiguous subsequences of fixed length w are classified in clusters using a similarity measure and these clusters receive a name (a symbol or a string of symbols). If we set $w = 2$ and classify not the sequences $x_i x_{i+1}$, but the normalized sequence

of the differences $x_{i+1} - x_i$, using the quantile of the normal distribution, we obtain the discretisation algorithm proposed by Keogh et al. [21]. But the biggest weakness of these methods which use a fixed length window is the sensibility to the noise. Therefore, the scale-space filtering method, which finds the boundaries of the subsequences having a persistent behavior over multiple degree of smoothing, seems to be more appropriate and must be considered as a first compulsory pre-processing phase.

Global feature calculation. During this step, one extracts various features from each sub-sequence as a whole. Typical global features include global maxima, global minima, means and standard deviation of the values of the sequence as well as the value of some specific point of the sequence, such as the value of the first or of the last point. Of course, it is possible that specific events will demand specific features, necessary for their description (e.g. the slope of the best-fitting line or the second real Fourier coefficient). The optimal set of global features is hard to be defined in advance, but as long as these features are simple descriptive statistics, they can be easily added or removed form the process.

Example 1. Consider a database containing daily price variations of a given stock. After the application of the first phase we obtain an ordered sequence of events. Each event has the form ($name$, v_1, v_2), where the name is one of the strings {*peak, flat, valley*} – we are interested only in three kinds of shapes - and v_1, v_2 represents the mean, respectively, the standard error – we chose only two features as determinant for the event. The statistics are calculated using daily prices, supposed to be subsequences of length $w = 12$.

2.2 Phase Two

During the second phase, we create a set of temporal rules inferred from the database of events, obtained in phase one. Two important steps can be defined here:

- Application of a first induction process, using the event database as training database, to obtain a set of *classification trees*. From each classification tree, the corresponding set of temporal rules is extracted.
- Application of a second inference process using the previously inferred temporal rules sets to obtain the final set of temporal meta-rules.

First induction process. There are different approaches for extracting rules from a set of events. Association Rules [6], Inductive Logic Programming [35], Classification Trees [20] are the most popular ones. For our methodology we selected the *classification tree approach.* It is a powerful tool used to predict memberships of cases or objects in the classes of a categorical dependent variable from their measurements on one or more predictor variables (or attributes). A classification tree is constructed by recursively partitioning a learning sample of data in which the class and the values of the predictor variables for each case are known. Each partition is represented by a node in

the tree. A variety of classification tree programs has been developed and we may mention QUEST [27], CART [4], FACT [28] and last, but not least, C4.5 [32]. Our option was the C4.5 like approach. The tree resulting by applying the C4.5 algorithm is constructed to minimize the observed error rate, using equal priors. This criterion seems to be satisfactory in the frame of sequential data and has the advantage to not favor certain events. To successively create the partitions, the C4.5 algorithm uses two forms of tests in each node: a standard test for discrete attributes, with one outcome ($A = x$) for each possible value x of the attribute A, and a binary test, for continuous attributes, with outcomes $A \leq z$ and $A > z$, where z is a threshold value. Finally, each path from the root to a leave corresponds to a rule representing a conjunction of tests outcomes (see the example from Fig. 1).

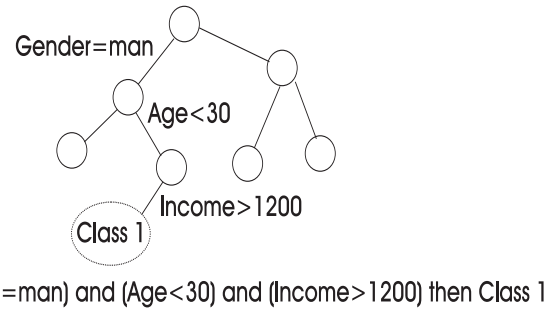

Fig. 1. Rule corresponding to a path from the root to the leave "Class 1", expressed as a conjunction of three outcome tests implying each a different attribute

Before to apply the decision tree algorithm to a database of events, an important problem has to be solved first: establishing the training set. An n-tuple in the training set contains $n-1$ values of the predictor variables (or attributes) and one value of the categorical dependent variable, which represents the class. There are two different approaches on how the sequence that represents the classification (the values of the categorical dependent variable) is obtained. In a supervised methodology, an expert gives this sequence. The situation becomes more difficult when there is no prior knowledge about the possible classifications. Suppose that, following the example we gave, we are interested in seeing if a given stock value depends on other stock values. As the dependent variable (the stock price) is not categorical, it cannot represent a valid classification used to create a classification tree. The solution is to use the sequence of the names of events extracted from the continuous time series as sequence of classes.

Let us suppose that we have k sequences representing the predictor variables $q_1, q_2, \ldots, q_k$. Each $q_{ij},\ i = 1, \ldots, k,\ j = 1, \ldots, n$ is the name of an event (*Remark:* we consider a simplified case, with no feature as predictor variable, but without influence on the following rationing). We have also a

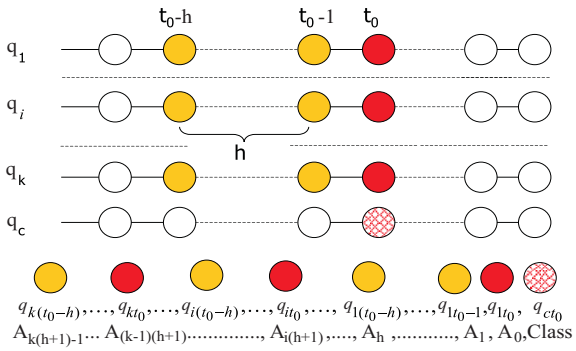

Fig. 2. Graphical representation of the first tuple and the list of corresponding attributes

sequence $q_c = q_{c1}, \ldots, q_{cn}$ representing the classification. The training set will be constructed using a procedure depending on three parameters. The first, t_0, represents a time instant considered as *present time*. Practically, the first tuple contains the class q_{ct_0} and there is no tuple in the training set containing an event that starts after time t_0. The second, t_p, represents a time interval and controls the further back in time class $q_{c(t_0-t_p)}$ included in the training set. Consequently, the number of tuples in the training set is t_p+1. The third parameter, h, controls the influence of the past events $q_{i(t-1)}, \ldots, q_{i(t-h)}$ on the actual event q_{it}. This parameter ($h$istory) reflects the idea that the class q_{ct} depends not only on the events at time t, but also on the events occurred before time t. Finally, each tuple contains $k(h+1)$ events (or values for $k(h+1)$ attributes, in the terminology of classification trees) and one class value (see Fig. 2). The first tuple is $q_{ct_0}, q_{1t_0}, \ldots, q_{1(t_0-h)}, \ldots, q_{k(t_0-h)}$ and the last $q_{c(t_0-t_p)}, q_{1(t_0-t_p)}, \ldots, q_{k(t_0-t_p-h)}$. To adopt this particular strategy for the construction of the training set, we made an assumption: the events q_{ij}, $i = 1, \ldots, k$, j a fixed value, occur all at the same time instant. The same assumption allows us to solve another implementation problem: the time information is not processed during the classification tree construction, (time is not a predictor variable), but the temporal dimension must be captured by the temporal rules. The solution we chose to encode the temporal information is to create a map between the index of the attributes (or predictor variables) and the order in time of the events. The $k(h+1)$ attributes are indexed as $\{A_0, A_1, \ldots, A_h, \ldots, A_{2h}, \ldots A_{k(h+1)-1}\}$. As we can see in Fig. 2, in each tuple the values of the attributes from the set $\{A_0, A_{h+1}, \ldots, A_{(k-1)(h+1)}\}$ represent events which occur at the same time moment as the class event, those of the set $\{A_1, A_{h+2}, \ldots, A_{(k-1)(h+1)+1}\}$ represent events which occur one time moment before the same class event, and so on. Let be $\{i_0, \ldots, i_m\}$ the set of indexes of the attributes that appear in the body of the rule (i.e. the rule has the form

$$\textit{If } (A_{i_0} = e_0) \textit{ and } (A_{i_1} = e_1) \textit{ and } \ldots \textit{ and } (A_{i_m} = e_m) \textit{ Then Class } e,$$

where e_{i_j} are events from the sequences $\{q_1, \ldots, q_k\}$ and e is an event from the sequence q_c. If t represents the time instant when the event in the head of the rule occurs, then an event from the rule's body, corresponding to the attribute A_{i_j}, occurred at time $t - \bar{i}_j$, where $\bar{i}_j$ means i *modulo* $(h+1)$.

Example 2. After the application of the first induction process, on the same stock prices database, we may obtain a temporal rule of the form: *If, during two consecutive days, the shape of the stock price had a* ***peak*** *form, with a mean less than* ***a*** *and a standard error greater than* ***b****, then, with a confidence of* ***c%****, three days later the stock price variation will present a* ***flat****.*

Second inference process. Different classification trees, constructed from different training sets, generate finally different sets of temporal rules. The process that tries to infer temporal meta-rules from these sets of local rules is derived from the rules pruning strategy used by C4.5 system. Because this strategy may be theoretically applied not only to the rules generated by C4.5 algorithm, but to all rules having the form of a general Horn clause, a modelling process of our methodology, at an abstract level, looks not only feasible, but also necessary. To obtain an abstract view of temporal rules we propose a formalism based on first-order temporal logic. This formalism allows not only to model the main concepts used by the algorithms applied during the different steps of the methodology, but also to give a common frame to many of temporal rules extraction techniques, mentioned in the literature.

3 Formalism of Temporal Rules

Time is ubiquitous in information systems, but the mode of representation/perception varies in function of the purpose of the analysis [7, 12]. Firstly, there is a choice of a *temporal ontology*, which can be based either on *time points* (instants) or on *intervals* (periods). Secondly, time may have a *discrete* or a *continuous* structure. Finally, there is a choice of *linear* vs. *nonlinear* time (e.g. acyclic graph). For our methodology, we chose a temporal domain represented by linearly ordered discrete instants. This is imposed by the discrete representation of all databases.

Definition 1 *A single-dimensional linearly ordered temporal domain is a structure $T_P = (T, <)$, where T is a set of time instants and "$<$" a linear order on T.*

Databases being first-order structures, the first-order logic represents a natural formalism for their description. Consequently, the first-order temporal logic expresses the formalism of temporal databases. For the purposes of our methodology we consider a restricted first-order temporal language L which contains only constant symbols $\{c, d, ..\}$, n-ary $(n \geq 1)$ function symbols $\{f, g, ..\}$, variable symbols $\{y_1, y_2, ...\}$, n-ary predicate symbols $(n \geq 1$, so no proposition symbols), the set of relational symbols $\{=, <, \leq, >, \geq\}$, the logical connective $\{\wedge\}$ and a temporal connective of the form X_k, $k \in \mathbf{Z}$,

where k strictly positive means *next k times*, k strictly negative means *last k times* and $k = 0$ means *now*.

The syntax of L defines terms, atomic formulae and compound formulae. The *terms* of L are defined inductively by the following rules:

T1. Each constant is a term.
T2. Each variable is a term.
T3. If $t_1, t_2, \ldots, t_n$ are terms and f is an n-ary function symbol then $f(t_1, \ldots, t_n)$ is a term.

The atomic formulae (or atoms) of L are defined by the following rules:

A1. If $t_1, \ldots, t_n$ are terms and P is an n-ary predicate symbol then $P(t_1, \ldots, t_n)$ is an atom.
A2. If $t_1,\ t_2$ are terms and ρ is a relational symbol then $t_1 \rho\, t_2$ is an atom (also called relational atom).

Finally, the (compound) formulae of L are defined inductively as follow:

F1. Each atomic formula is a formula.
F2. If $p,\ q$ are formulae then $(p \wedge q),\ X_k p$ are formulae.

A Horn clause is a formula of the form $B_1 \wedge \cdots \wedge B_m \rightarrow B_{m+1}$ where each B_i is a positive (non-negated) atom. The atoms $B_i,\ i = 1, \ldots, m$ are called implication clauses, whereas B_{m+1} is known as the implicated clause. Syntactically, we cannot express Horn clauses in our language L because the logical connective $\rightarrow$ is not defined. However, to allow the description of rules, which formally look like a Horn clause, we introduce a new logical connective, $\mapsto$, which practically will represent a rewrite of the connective $\wedge$. Therefore, a formula in L of the form $p \mapsto q$ is syntactically equivalent to the formula $p \wedge q$. When and under what conditions we may use the new connective, is explained in the next definitions.

Definition 2 *An event (or temporal atom) is an atom formed by the predicate symbol E followed by a bracketed n-tuple of terms* $(n \geq 1)$ $E(t_1, t_2, \ldots, t_n)$. *The first term of the tuple,* t_1*, is a constant symbol representing the name of the event and all others terms are expressed according to the rule T3* ($t_i = f(t_{i1}, \ldots, t_{ik_i})$). *A short temporal atom (or the event's head) is the atom* $E(t_1)$.

For each constant symbol t used as an event name, two other constant symbols, *start_t* and *stop_t*, are included in our language L. Consequently, for each temporal atom $E(t_1, t_2, \ldots, t_n)$, two temporal atoms, $E(start\_t_1, t_2, \ldots, t_n)$ and $E(stop\_t_1, t_2, \ldots, t_n)$, are defined.

Definition 3 *A constraint formula for the event* $E(t_1, t_2, \ldots t_n)$ *is a conjunctive compound formula,* $E(t_1, t_2, \ldots t_n) \wedge C_1 \wedge C_2 \wedge \cdots \wedge C_k$*, where each* C_j *is a relational atom. The first term of* C_j *is one of the terms* t_i*,* $i = 1 \ldots n$ *and the second term is a constant symbol.*

For a short temporal atom $E(t_1)$, the only constraint formula that is permitted is $E(t_1) \wedge (t_1 = c)$. We denote such constraint formula as *short constraint formula*.

Definition 4 *A temporal rule is a formula of the form* $H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$, *where* H_{m+1} *is a short constraint formula and* H_i *are constraint formulae, prefixed by the temporal connectives* X_{-k}, $k \geq 0$. *The maximum value of the index* k *is called the time window of the temporal rule.*

As a consequence of the Definition 4, a conjunction of constraint formulae $H_1 \wedge H_2 \wedge \cdots \wedge H_n$, each formula prefixed by temporal connectives X_{-k}, $k \geq 0$, may be rewritten as $H_{\sigma(1)} \wedge \cdots \wedge H_{\sigma(n-1)} \mapsto H_{\sigma(n)}$, — σ being a permutation of $\{1..n\}$ — only if there is a short constraint formula $H_{\sigma(n)}$ prefixed by X_0.

Remark. The reason for which we did not permit the expression of the implication connective in our language is related on the truth table for a formula $p \rightarrow q$: even if p is false, the formula is still true, which is unacceptable for a temporal rationing of the form *cause*$\rightarrow$ *effect*.

If we change in Definition 2 the conditions imposed to the terms $t_i, i = 1...n$ to "each term t_i is a variable symbol", we obtain the definition of a temporal atom template. We denote a such template as $E(y_1, \ldots, y_n)$. Following the same rationing, a constraint formula template for $E(y_1, \ldots, y_n)$ is a conjunctive compound formula, $C_1 \wedge C_2 \wedge \cdots \wedge C_k$, where the first term of each relational atom C_j is one of the variables y_i, $i = 1 \ldots n$. Consequently, a short constraint formula template is the relational atom $y_1 = c$. Finally, by replacing in Definition 4 the notion "constraint formula" with "constraint formula template" we obtain the definition of a temporal rule template. Practically, the only formulae constructed in L are temporal atoms, constraint formulae, temporal rules and the corresponding templates.

The semantics of L is provided by an interpretation I over a domain D (in our formalism, D is always a linearly ordered domain). The interpretation assigns an appropriate meaning over D to the (non-logical) symbols of L. More precisely I assigns a meaning to the symbols of L as follows:

- for an n-ary predicate symbol P, $n \geq 1$, the meaning $I(P)$ is a function $D^n \rightarrow B$, where B is the set $\{true, false\}$,
- for an n-ary function symbol, f, $n \geq 1$, the meaning $I(f)$ is a function $D^n \rightarrow D$,
- for a constant symbol c the meaning $I(c)$ is an element of D,
- for a variable symbol y the meaning $I(y)$ is a element of D.

The interpretation I is extended to arbitrary terms as $I(f(t_1, \ldots, t_n)) = I(f)(I(t_1), \ldots, I(t_n))$. For atomic formulae and compound formulae, we have:

- If $t_1, \ldots, t_n$ are terms and P is an n-ary predicate symbol then $I(P(t_1, \ldots, t_n)) = true$ iff $I(P)(I(t_1), \ldots, I(t_n)) = true$.
- If t_1, t_2 are terms and ρ is a relational symbol then $I(t_1 \rho\, t_2) = true$ iff $I(t_1) \rho I(t_2)$.

- If p, q are formulae then $I(p \wedge q) = true$ iff $I(p) = true$ and $I(q) = true$.

Usually, the domain D is imposed during the discretisation phase, which is a pre-processing phase used in almost all knowledge extraction methodologies. Based on Definition 2, an event can be seen as a labelled (constant symbol t_1) sequence of points extracted from raw data and characterized by a finite set of features (terms $t_2, \cdots, t_n$). Let be D_e the set containing all the strings used as event names. We will extend this set by adding, for each $e \in D_e$, the strings *start_e* and *stop_e*. Finally, the domain D is the union $D_e \cup D_f$, where D_e is the extended set of strings and D_f represents the union of all sub-domains corresponding to chosen features.

To define a first-order linear temporal logic based on L, we need a structure having a temporal dimension and capable to capture the relationship between a time moment and the interpretation I at this moment.

Definition 5 *Given L and a domain D, a (first order) linear time structure is a triple $M = (S, x, \boldsymbol{I})$, where S is a set of states, $x : \mathbf{N} \to S$ is an infinite sequence of states $(s_0, s_1, \ldots, s_n, \ldots)$ and $\boldsymbol{I}$ is a function that associates with each state s an interpretation $\boldsymbol{I}_s$ of all symbols from L.*

In the framework of linear temporal logic, the set of symbols is divided into two classes, the class of global symbols and the class of local symbols. Intuitively, a global symbol w has the same interpretation in each state, i.e. $\mathbf{I}_s(w) = \mathbf{I}_{s'}(w) = \mathbf{I}(w)$, for all $s, s' \in S$; the interpretation of a local symbol may vary, depending on the state at which is evaluated. The formalism of temporal rules assumes that all function symbols (including constants) and all relational symbols are global, whereas the predicate symbols and variable symbols are local. Consequently, as the temporal atoms, constraint formulae, temporal rules and the corresponding templates are expressed using the predicate symbol E or the variable symbols y_i, the meaning of truth for these formulae depend on the state at which are evaluated. Given a first order time structure M and a formula p, we denote the instant i (or equivalently, the state s_i) for which $\mathbf{I}_{s_i}(p) = true$ by $i \models p$, i.e. at time instant i the formula p is true. Therefore, $i \models E(t_1, \ldots, t_n)$ means that at time i an event with the name $\mathbf{I}(t_1)$ and characterized by the global features $\mathbf{I}(t_2), \ldots, \mathbf{I}(t_n)$ occurs. Using this definition, we can also define:

- $i \models E(start\_t_1, \ldots, t_n)$ iff $i \models E(t_1, \ldots, t_n)$ and $(i-1) \not\models E(t_1, \ldots, t_n)$,
- $i \models E(stop\_t_1, \ldots, t_n)$ iff $i \models E(t_1, \ldots, t_n)$ and $(i+1) \not\models E(t_1, \ldots, t_n)$

Concerning the event template $E(y_1, \ldots, y_n)$, the interpretation of the variable symbols y_j at the state s_i, $\mathbf{I}_{s_i}(y_j)$, is chosen such that $i \models E(y_1, \ldots, y_n)$ for every time moment i. Because

- $i \models p \wedge q$ if and only if $i \models q$ and $i \models q$, and
- $i \models X_k p$ if and only if $i + k \models p$,

a constraint formula (template) is true at time i if and only if all relational atoms are true at time i and $i \models E(t_1, \ldots, t_n)$, whereas a temporal rule (template) is true at time i if and only if $i \models H_{m+1}$ and $i \models (H_1 \wedge \cdots \wedge H_m)$.

Now suppose that the following assumptions are true:

A. For each formula p in L, there is an algorithm that calculates the value of the interpretation $\mathbf{I}_s(p)$, for each state s, in a finite number of steps.
B. There are states (called incomplete states) that do not contain enough information to calculate the interpretation for all formulae defined at these states.
C. It is possible to establish a measure, (called *general interpretation*) about the degree of truth of a compound formula along the entire sequence of states $(s_0, s_1, \ldots, s_n, \ldots)$.

The first assumption express the calculability of the interpretation **I**. The second assumption express the situation when only the body of a temporal rule can be evaluated at time moment i, but not the head of the rule. Therefore, for the state s_i, we cannot calculate the interpretation of the temporal rule and the only solution is to estimate it using a general interpretation. This solution is expressed by the third assumption. (*Remark:* The second assumption violates the condition about the existence of an interpretation in each state s_i, as defined in Definition 5. But it is well known that in data mining sometimes data are incomplete or are missing. Therefore, we must modify this condition as "***I*** *is a function that associates with* `almost` *each state s an interpretation* $\boldsymbol{I}_s$ *of all symbols from L* ").

However, to ensure that this general interpretation is well defined, the linear time structure must present some property of consistency. Practically, this means that if we take any sufficiently large subset of time instants, the conclusions we may infer from this subset are sufficiently close from those inferred from the entire set of time instants. Therefore,

Definition 6 *Given L and a linear time structure M, we say that M is a consistent time structure for L if, for every atomic formula p, the limit* $supp(p) = \lim_{n\to\infty} \frac{\#A}{n}$ *exists, where* $A = \{i \in \{0, \ldots, n\} | i \models p\}$ *and # means "cardinality". The notation supp(p) denotes the support (of truth) of p.*

Now we define the general interpretation for an n-ary predicate symbol P as:

Definition 7 *Given L and a consistent linear time structure M for L, the general interpretation* I_G *for an n-ary predicate P is a function* $D^n \rightarrow \{true\} \times [0, 1]$*, such that, for each n-tuple of terms* $\{t_1, \ldots, t_n\}$*,* $I_G(P(t_1, \ldots, t_n)) = (true, supp(P(t_1, \ldots, t_n))$.

The general interpretation is naturally extended to constraint formulae, temporal rules and the corresponding templates. There is another useful measure, called *confidence*, but available only for temporal rules (templates). This measure is calculated as a limit ratio between the number of certain applications

(time instants where both the body and the head of the rule are true) and the number of potential applications (time instants where only the body of the rule is true). The reason for this choice is related to the presence of incomplete states, where the interpretation for the implicated clause cannot be calculated.

Definition 8 *The confidence of a temporal rule (template)* $H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$ *is the limit* $\lim_{n\to\infty} \frac{\#A}{\#B}$, *where* $A = \{i \in \{0,\ldots,n\} | i \models H_1 \wedge \cdots \wedge H_m \wedge H_{m+1}\}$ *and* $B = \{i \in \{0,\ldots,n\} | i \models H_1 \wedge \cdots \wedge H_m\}$.

For different reasons, (the user has not access to the entire sequence of states, or the states he has access to are incomplete), the general interpretation cannot be calculated. A solution is to estimate I_G using a finite linear time structure, i.e. a model.

Definition 9 *Given* L *and a consistent time structure* $M = (S, x, \boldsymbol{I})$, *a model for* M *is a structure* $\tilde{M} = (\tilde{T}, \tilde{x})$ *where* $\tilde{T}$ *is a finite temporal domain* $\{i_1,\ldots,i_n\}$, $\tilde{x}$ *is the subsequence of states* $\{x_{i_1},\ldots,x_{i_n}\}$ *(the restriction of* x *to the temporal domain* $\tilde{T}$*) and for each* $i_j, j = 1,\ldots,n$, *the state* x_{i_j} *is a complete state.*

Now we may define the estimator for a general interpretation:

Definition 10 *Given* L *and a model* $\tilde{M}$ *for* M, *an estimator of the general interpretation for an* n*-ary predicate* P, $\tilde{I}_G(P)$, *is a function* $D^n \to \{true\} \times [0,1]$, *assigning to each atomic formula* $p = P(t_1,\ldots,t_n)$ *the value* true *with a support equal to the ratio* $\frac{\#A}{\#\tilde{T}}$, *where* $A = \{i \in \tilde{T} | i \models p\}$. *The notation* $supp(p, \tilde{M})$ *will denote the estimated support of* p, *given* $\tilde{M}$.

Once again, the estimation of the confidence for a temporal rule (template) is defined as:

Definition 11 *Given a model* $\tilde{M} = (\tilde{T}, \tilde{x})$ *for* M, *the estimation of the confidence for the temporal rule (template)* $H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$ *is the ratio* $\frac{\#A}{\#B}$, *where* $A = \{i \in \tilde{T} | i \models H_1 \wedge \cdots \wedge H_m \wedge H_{m+1}\}$ *and* $B = \{i \in \tilde{T} | i \models H_1 \wedge \cdots \wedge H_m\}$.

4 Methodology Versus Formalism

As it was extensively presented in Sect. 2, the methodology for temporal rules extraction may be structured in two phases. During the first phase one transform sequential raw data into sequences of events. Practically, this means to establish the set of events, identified by names, and the set of features, common for all events.

In the frame of our formalism, during this phase we establish the set of temporal atoms which can be defined syntactically in L. For this we start by defining the first-order temporal language L. Considering as raw data the database described in Example 1, we include in L a 3-ary predicate symbol E, three variable symbols $y_i, i = 1..3$, two 12-ary function symbols f and g, two sets of constant symbols – $\{d_1, \ldots, d_6\}$ and $\{c_1, \ldots, c_n\}$ – and the usual set of relational symbols and logical(temporal) connectives. As we showed in the above example and according to the syntactic rules of L, an event is defined as $E(d_i, f(c_{j_1}, \ldots, c_{j_{12}}), g(c_{k_1}, \ldots, c_{k_{12}}))$, whereas an event template is defined as $E(y_1, y_2, y_3)$. Also provided during this phase is the semantics of L. Firstly, the domain $D = D_e \cup D_f$ (see Sect. 3) is defined. According to the results of the discretisation algorithm applied on raw data from the cited example, the domain D_e is defined as {*peak, start_peak, stop_peak, flat, start_flat, stop_flat, valley, start_valley, stop_valley*}. During the step *global features calculation*, two features – the mean and the standard error – were selected to capture the continuous aspect of the events. Consequently, the domain $D_f = \Re^+$, as the stock prices are positives real numbers and the features are statistical functions.

Secondly, a linear time structure M, i.e. a triple $(S, x, \mathbf{I})$ (see Def. 5) is specified. The database of events, obtained after the first phase of the methodology, contains tuples with three values, (v_1, v_2, v_3). For a tuple with a recording index i, the first value expresses the name of the event – *peak*, *flat*, *valley* – which occurs at time moment i and the two other values express the result of the two features. Therefore, we define a state s as a triple (v_1, v_2, v_3), the set S as the set of all tuples from database and the sequence x as the ordered sequence of tuples in database (see Table 1).

Table 1. The first nine states of the linear time structure M (example)

Index	State	Index	State	Index	State
1	$(peak, 10, 1.5)$	4	$(flat, 1, 0.5)$	7	$(valley, 15, 1.9)$
2	$(peak, 10, 1.5)$	5	$(flat, 1, 0.5)$	8	$(flat, 3, 1.1)$
3	$(peak, 14, 2.2)$	6	$(flat, 1, 0.5)$	9	$(peak, 12, 1.2)$

At this stage the interpretation of all symbols (global and local symbols) can be defined. For the global symbols (function symbols and relational symbols), the interpretation is quite intuitive. Therefore, the meaning $\mathbf{I}(d_j)$ is an element of D_e, the meaning $\mathbf{I}(c_j),\ j = 1..n$, is a positive real number, whereas the meaning $\mathbf{I}(f)$, respectively $\mathbf{I}(g)$, is the function $f : D_f^{12} \to D_f$, $f(\mathbf{x}) = \overline{\mathbf{x}}$, respectively the function $g : D_f^{12} \to D_f$, $g(\mathbf{x}) = \mathrm{se}(\mathbf{x})$ – we used the standard notations in statistics for the mean and standard error estimators.

The interpretation of a local symbol (the variable symbols y_i and the predicate symbol E) depends on the state at which is evaluated. According

Table 2. The temporal atoms evaluated *true* at the first nine states of M (example)

State	Temporal atom
1	$E(peak, 10, 1.5)$, $E(start\_peak, 10, 1.5)$
2	$E(peak, 10, 1.5)$, $E(stop\_peak, 10, 1.5)$
3	$E(peak, 14, 2.2)$, $E(start\_peak, 14, 2.2)$, $E(stop\_peak, 14, 2.2)$
4	$E(flat, 1, 0.5)$, $E(start\_flat, 1, 0.5)$
5	$E(flat, 1, 0.5)$
6	$E(flat, 1, 0.5)$, $E(stop\_flat, 1, 0.5)$
7	$E(valley, 15, 1.9)$, $E(start\_valley, 15, 1.9)$, $E(stop\_valley, 15, 1.9)$
8	$E(flat, 3, 1.1)$, $E(start\_flat, 3, 1.1)$, $E(stop\_flat, 3, 1.1)$
9	$E(peak, 12, 1.2)$, $E(start\_peak, 12, 1.2)$, $E(stop\_peak, 12, 1.2)$

to the assumption A (see Sect. 3), the function $\mathbf{I}_{s_i}(E)$ defined on D^3 with values in $B = \{true, false\}$ is provided by a finite algorithm. This algorithm will receive at input at least the state s_i and will provide at output one of the values from B. Therefore, the interpretation of $E(t_1, t_2, t_3)$ evaluated at s_i is defined as:

Algorithm 1 ***Temporal atom evaluation***

Consider the state $s_i = (v_1, v_2, v_3)$
If $(\mathbf{I}_{s_i}(t_1) = v_1)$ *and* $(\mathbf{I}_{s_i}(t_2) = v_2)$ *and* $(\mathbf{I}_{s_i}(t_3) = v_3)$
Then $\mathbf{I}_{s_i}(E(t_1, t_2, t_3)) = true$
Else $\mathbf{I}_{s_i}(E(t_1, t_2, t_3)) = false$

Finally, the interpretation of the variable symbol y_j at the state s_i is given by $\mathbf{I}_{s_i}(y_j) = v_j, j = 1..3$, which satisfies the condition imposed to the interpretation of temporal atom template (see Sect. 3), which is $\mathbf{I}_{s_i} E(y_1, y_2, y_3) = true$ for each state s_i. Having well-defined the language L, the syntax and the semantics of L, as well as the linear time structure M, we can construct the temporal atoms evaluated as *true* at time moment i (see Table 2)

During the second phase of the methodology, we generate a set of temporal rules inferred from the database of events, obtained in phase one. The first induction process consists in creating classification trees, each based on a different training set. In the frame of our formalism, choosing a training set is equivalent to choose a model $\tilde{M}$ for the linear time structure M. All the states from these models are complete states, because the algorithm which construct the tree must know, for each time moment, the set of predictor events and the corresponding dependent event. Once the classification tree constructed, the test contained in each node becomes a relational atom and the set of all relational atoms situated on a path from root to a leaf become a constraint formula template. The variable symbols y_i included in the template are generated by the following rule:

- if the attribute concerned by the test is related to the event name, it is replaced by y_1
- if the attribute concerned by the test is related to the feature mean (respectively standard error), it is replaced by y_2 (respectively y_3).

The constraint formula template becomes a temporal rule template by adding temporal connectives according to the procedure which links a temporal dimension to a rule generated by C4.5 algorithm. Finally, the confidence of temporal rule template is calculated according to the Definition 11.

Remark: The values of the categorical dependent variable, or the classes, may be obtained either in a supervised mode (e.g. given by an expert) or in an unsupervised mode (e.g. the names of the events). In the last case, we may restrict the possible values to the set $\{start\_event_1, stop\_event_1, \ldots, start\_event_n, stop\_event_n\}$).

To exemplify the induction process, from training set to temporal rule templates, consider the following model $\tilde{M} = \{s_1, \ldots, s_{100}\}$. According to the procedure for training set selection (see Sect. 2), in a first step we must indicate the sequences q_i, $i = 1..k$, of predictor variables and the sequence q_c of class values. The information on sequences q_i is extracted from the structure of the states from the model $\tilde{M}$: given the state $s_i = (v_1, v_2, v_3)$, $1 \leq i \leq 100$, we define $q_{ji} = v_j$, $j = 1..3$. Therefore, q_1 is the sequence of the event names, q_2 is the sequence of the mean values and q_3 is the sequence of the standard error values. As there is no predefined classification (unsupervised mode), the sequence q_c is defined as $q_{ci} = q_{1i}, i = 1..100$. The next step consists in defining the parameters t_0, t_p and h, which are set as $t_0 = 100$, $t_p = 96$ and $h = 3$ (the methodology for finding the optimal value of the parameter h is based on the analysis of classification errors and is described in Cotofrei and Stoffel [10]). Concerning the tuples from the training set, there is a minor difference compared with the procedure from Sect. 2 – the sequence of class values (q_c)

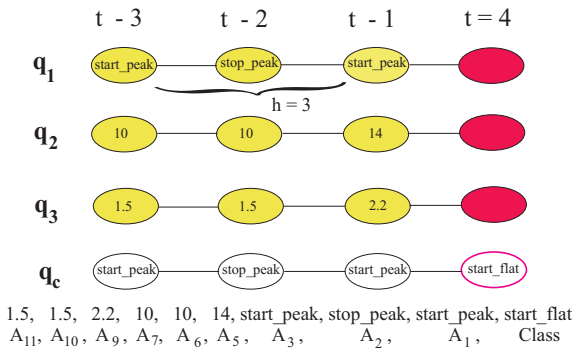

Fig. 3. Graphical representation of the last tuple of the training set based on states from Table 1 and defined by the parameters $t_0 = 100$, $t_p = 96$ and $h = 3$ (inclusive the list of corresponding attributes)

being the same as one of the sequence of predictor variables (q_1), we can not include in a tuple which contains the class value q_{ct} the predictor values q_{1t}, q_{2t} and q_{3t}. In other words, we can not accept that the same event occurred at time t to be simultaneously on the left and on the right side of a temporal rule. As we can see in Fig. 3, a tuple contains now $k \cdot h = 9$ predictor values instead of $k(h+1) = 12$, and there is no attribute having an index i such that i *modulo* $(h+1)$ to be 0. Suppose now that one of the rule generated by C4.5 algorithm using the previous defined training set has the form

If (A_3=*start_peak*) *and* ($A_7 < 11$) *and* (A_1=*start_peak*) *Then Class start_valley*

By convention, the event in the head of the rule occurs always at time moment $t = 0$, so an event from the body of the rule, corresponding to the attribute A_i, occurs at time moment $-(i \ modulo \ 4)$. By applying this observation and the convention on how to use symbol variables in a temporal rule, we obtain the following temporal rule template

$$X_{-3}(y_1 = start\_peak) \wedge X_{-3}(y_2 < 11) \\ \wedge X_{-1}(y_1 = start\_peak) \mapsto X_0(y_1 = start\_valley)$$

The induction process is repeatedly applied on different models $\tilde{M}_i$ of the time structure M, which will generate in the end different sets of temporal rule templates (see Table 3). It is possible to obtain the same template in two different sets, but with different confidence, or templates with the same head, but with different bodies.

The second inference process is designed to obtain temporal meta-rules, which are temporal rule templates in accordance with the Definition 4, but supposed to have a small variability of the estimated confidence among different models. Therefore, a temporal meta-rule may be applied with the same confidence in any state, complete or incomplete. To obtain such temporal

Table 3. Different temporal rule templates extracted from two models $\tilde{M}$ using the induction process (example)

Model	Temporal Rule Templates
$s_1 \dots s_{100}$	$X_{-3}(y_1 = start\_peak) \wedge X_{-3}(y_2 < 11) \wedge$ $\wedge X_{-1}(y_1 = start\_peak) \mapsto X_0(y_1 = start\_valley)$ $X_{-2}(y_1 = start\_peak) \wedge X_{-2}(y_3 < 1.1) \wedge$ $\wedge X_{-1}(y_1 = stop\_flat) \mapsto X_0(y_1 = start\_valley)$ . . . . .
$s_{300} \dots s_{399}$	$X_{-2}(y_1 = peak) \wedge X_{-2}(y_3 < 1.1) \wedge$ $\wedge X_{-2}(y_2 \geq 12.3) \mapsto X_0(y_1 = start\_valley)$ $X_{-4}(y_1 = stop\_peak) \wedge X_{-3}(y_1 = start\_flat) \wedge X_{-3}(y_2 >= 3.2) \wedge$ $\wedge X_{-3}(y_3 < 0.4) \wedge X_{-1}(y_1 = stop\_flat) \mapsto X_0(y_1 = start\_peak)$ . . . . .

rules, we apply strategies which cut irrelevant relational atoms, according with some criterions, from the implication clauses of temporal rules templates obtained during the first induction process. The process of inferring temporal meta-rules is related to a new approach in data mining, called *higher order mining*, i.e. mining from the results of previous mining runs. According to this approach, the rules generated by the first induction process are first order rules and those generated by the second inference process (i.e. temporal meta-rules) are higher order rules. The formalism we proposed does not impose what methodology to use to discover first order temporal rules. As long as these rules satisfy the syntactic form described in Definition 4, the strategy (including algorithms, criterions, statistical methods) developed to infer temporal meta-rules might be applied.

5 Temporal Meta-Rules

Suppose that for a given model $\tilde{M}$ we dispose of a set of temporal rules templates, extracted from the corresponding classification tree. It is very likely that some temporal rules templates contain implication clauses that are irrelevant, i.e. after their deletion, the general interpretation of the templates remain unchanged (*Remark*: in the following, by the notion "implication clause" we consider a relational atom prefixed by the temporal connective X_{-k}). In the frame of a consistent time structure M, it is obviously that we cannot delete an implication clause from a temporal rule template (denoted TR) if the resulting template (noted TR^{-}) has a lower confidence. But for a given model $\tilde{M}$, we calculate an estimate (denoted $co(TR, \tilde{M})$) of the true confidence (denoted $co(TR)$). Because this estimator has a binomial distribution, we may establish a confidence interval for $co(TR)$ and, consequently, we accept to delete an implication clause from TR if and only if the lower confidence limit of $co(TR^{-}, \tilde{M})$ is greater than the lower confidence limit of $co(TR, \tilde{M})$.

The estimator $co(TR, \tilde{M})$ being the ratio $\#A/\#B$, (see Def. 11), a confidence interval for this value is constructed using a normal distribution depending on $\#A$ and $\#B$ (more precisely, the normal distribution has mean $\pi = \#A/\#B$ and variance $\sigma^2 = \pi(1-\pi)/\#B$). The lower limit of the interval is $L_\alpha(A, B) = \pi - z_\alpha \sigma$, where z_α is a quantile of the normal distribution for a given confidence level α. The algorithm which generalize a single temporal rule template TR, by deleting a single implication clause, is presented in the following:

Algorithm 2 ***Generalization 1-delete***

Step 1 Let $TR = H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$ *be a temporal rule template. Let* $\aleph = \cup\{C_j\}$*, where* C_j *are all the implication clauses that appear in the body of the template. Rewrite TR, by an abuse of notation, as* $\aleph \mapsto H_{m+1}$*. If* $n = \#\aleph$*, denote by* $C_1, \ldots, C_n$ *the list of all implication clauses from* $\aleph$*.*

Step 2 For each $i = 1, \ldots, n$ *do*

$\aleph^- = \aleph - C_i, \quad TR_i^- = \aleph^- \mapsto H_{m+1}$
$A = \{i \in \tilde{T} | i \models \aleph \wedge H_{m+1}\}, B = \{i \in \tilde{T} | i \models \aleph\}$
$A^- = \{i \in \tilde{T} | i \models \aleph^- \wedge H_{m+1}\}, B^- = \{i \in \tilde{T} | i \models \aleph^-\}$
$co(TR, \tilde{M}) = \#A/\#B,\ co(TR_i^-, \tilde{M}) = \#A^-/\#B^-$
If $L_\alpha(A, B) \leq L_\alpha(A^-, B^-)$ *then store* TR_i^-

Step 3 Keep only the generalized temporal rule template TR_i^- *for which* $L_\alpha(A^-, B^-)$ *is maximal.*

The core of the algorithm is the **Step 2**, where the sets used to estimate the confidence of the initial template, TR, and of the generalized template, TR^-, i.e. A, B, A^- and B^-, are calculated. The complexity of this algorithm is linear in n. Using the criterion of lower confidence limit, (or LCL), we define the temporal meta-rule inferred from TR as the temporal rule template with a maximum set of implication clauses deleted from $\aleph$ and having the maximum lower confidence limit greater than $L_\alpha(A, B)$. An algorithm designed to find the largest subset of implication clauses that can be deleted will have an exponential complexity. A possible solution is to use the Algorithm 2 in successive steps until no more deletion is possible, but without having the guarantee that we will get the global maximum.

As example, consider the first temporal rule template from Table 3 and suppose that $\#A = 20$ and $\#B = 40$ (giving an estimate $co(TR, \tilde{M}) = 0.5$, with $L_{0.95}(0.5) = 0.345$). Looking at Table 4, we find two implication clauses which could be deleted (the first and the second) with a maximum $L_\alpha(A^-, B^-)$ given by the second clause. As a remark, by deleting the first implication clause, the resulting template has an estimate of the confidence (0.489) less than of the original template (0.5), but a lower confidence limit 0.349 greater than $L_{0.95}(0.5)$, which allows the deletion operation.

Table 4. Parameters calculated in Step 2 of the Algorithm 2 by deleting one implication clause from the template $X_{-3}(y_1 = start\_peak) \wedge X_{-3}(y_2 < 11) \wedge X_{-1}(y_1 = start\_peak) \mapsto X_0(y_1 = start\_valley)$

Deleted implication clause	$\#A^-$	$\#B^-$	$co(TR_i^-, \tilde{M})$	$L_\alpha(A^-, B^-)$
$X_{-3}(y_1 = start\_peak)$	24	49	0.489	0.349
$X_{-3}(y_2 < 11)$	30	50	0.60	0.464
$X_{-1}(y_1 = start\_peak)$	22	48	0.458	0.317

If we apply again the Algorithm 2 on the template

$$X_{-3}(y_1 = start\_peak) \wedge X_{-1}(y_1 = start\_peak) \mapsto X_0(y_1 = start\_valley)$$

(denoted TR^-), we find that no other implication clause can be deleted, i.e. TR^- is the temporal meta rule according to the criterion LCL inferred from the temporal rule template

$$X_{-3}(y_1 = start\_peak) \wedge X_{-3}(y_2 < 11) \wedge \\ X_{-1}(y_1 = start\_peak) \mapsto X_0(y_1 = start\_valley).$$

Suppose now that we dispose of two models, $\tilde{M}_1 = (\tilde{T}_1, \tilde{x}_1)$ and $\tilde{M}_2 = (\tilde{T}_2, \tilde{x}_2)$, and for each model we have a set of temporal rule templates with the same implicated clause H (sets denoted S_1, respectively S_2). Let S be a subset of the union $S_1 \cup S_2$. If $TR_j \in S$, $j = 1, \ldots, n$, $TR_j = H_1 \wedge \cdots \wedge H_{m_j} \mapsto H$, then consider the sets

$$A_j = \{i \in \tilde{T}_1 \cup \tilde{T}_2 | i \models H_1 \wedge \ldots \wedge H_{m_j} \wedge H\}, \mathbf{A} = \cup A_j,$$
$$B_j = \{i \in \tilde{T}_1 \cup \tilde{T}_2 | i \models H_1 \wedge \ldots \wedge H_{m_j}\}, \mathbf{B} = \cup B_j,$$
$$\mathbf{C} = \{i \in \tilde{T}_1 \cup \tilde{T}_2 | i \models H\}.$$

The performance of the subset S can be summarized by the number of *false positives* (time instants where the implication clauses of each template from S are true, but not the clause H) and the number of *false negatives* (time instants where the clause H is true, but none of the implication clauses of the templates from S). Practically, the number of *false positives* is $fp = \#(\mathbf{B} - \mathbf{A})$ and the number of *false negatives* is $fn = \#(\mathbf{C} - \mathbf{B})$. The worth of the subset S of temporal rule templates is assessed using the Minimum Description Length Principle (MDLP)[34, 33]. This provides a basis for offsetting the accuracy of a theory (here, a subset of templates) against its complexity. The principle is simple: a Sender and a Receiver have both the same models $\tilde{M}_1$ and $\tilde{M}_2$, but the states from the models of the Receiver are incomplete states (the interpretation of the implicated clause cannot be calculated). The sender must communicate the missing information to the Receiver by transmitting a theory together with the exceptions to this theory. He may choose either a simple theory with a great number of exceptions or a complex theory with fewer exceptions. The MLD Principle states that the best theory will minimize the number of bits required to encode the total message consisting of the theory together with its associated exceptions. This is a particular instantiation of the MLDP, called two-part code version, which states that, among the set of candidate hypotheses $\mathcal{H}$, the best hypothesis to explain a set of data is one which minimizes the sum of the length, in bits, of the description of the hypothesis, and the length, in bits, of the description of the data encoded with the help of the hypothesis (which usually amounts to specifying the errors the hypothesis makes on the data). In the case where there are different hypotheses for which the sum attains its minimum, we select those with a minimum description length.

To encode a temporal rule template from S, we must specify its implication clauses (the implicated clause being the same for all rules, there is no need

to encoded it). Because the order of the implication clauses is not important, the number of required bits is reduced by $\kappa \log_2(m!)$, where m is the number of implication clauses and κ is a constant depending on encoding procedure. The number of bits required to encode the set S is the sum of encoding length for each template from S reduced by $\kappa \log_2(n!)$ (the order of the n templates from S is not important). The exceptions are encoded by indicating the sets *false positive* and *false negative*. If $b = \#\mathbf{B}$ and $N = \#(\tilde{T}_1 + \tilde{T}_2)$ then the number of bits required is $\kappa \log_2 \left(\binom{b}{fp} \right) + \kappa \log_2 \left(\binom{N-b}{fn} \right)$, because we have $\binom{b}{fp}$ possibilities to choose the *false positives* among the cases covered by the rules and $\binom{N-b}{fn}$ possibilities to indicate the *false negatives* among the uncovered cases. The total number of bits required to encode the message is then equal to *theory bits + exceptions bits*.

Using the criterion of MDLP, we define as temporal meta-rules inferred from a set of temporal rule templates (implying the same clause and extracted from at least two different models), the subset S that minimizes the total encoding length. An algorithm designed to extensively search this subset S has an exponential complexity, but in practice (and especially when $\#(S_1 + S_2) > 10$) we may use different non-optimal strategies (hill-climbing, genetic algorithms, simulated annealing), having a polynomial complexity.

For a practical implementation of an encoding procedure in the frame of the formalism, one use a notion from the theory of probability, i.e. the entropy. Given a finite set S, the entropy of S is defined as $\mathcal{I}(S) = -\sum_{v \in S} freq(v) \cdot \log_2(freq(v))$, where $freq(v)$ means the frequency of the element v in S. This measure attains its maximum when all frequencies are equal. Consider now a model $\tilde{M}$, characterized by the states $s_1, \ldots, s_n$, where each state s_i is defined by a m-tuple $(v_{i_1}, \ldots, v_{i_m})$. Based on these states consider the sets $A_j, j = 1..m$, where $A_j = \bigcup_{i=1..n} \{v_{i_j}\}$. Let be a temporal rule template inducted from the model $\tilde{M}$ and let be $X_{-k}(y_j \ \rho \ c)$ an implication clause from this template, with $j \in \{1 \ldots m\}$ and ρ a relational symbol. We define the encoding length for $X_{-k}(y_j \ \rho \ c)$ to be $\mathcal{I}(A_j)$. The encoding length of a temporal rule template having k implication clauses is thence equal with $\log_2(k)$ plus the sum of encoding length for each clause, reduced by $\log_2(k!)$ (order is not important), but augmented with $\log_2(m \cdot h_{max})$, where h_{max} is the time window of the template. The last quantity expresses the encoding length for the temporal dimension of the rule. Finally, the encoding length of q temporal rule templates is $\log_2(q)$ plus the sum of encoding length for each template, reduced by $\log_2(q!)$ (order is not important), whereas the encoding length of the exceptions is given by $\log_2 \left(\binom{b}{fp} \right) + \log_2 \left(\binom{N-b}{fn} \right)$.

As example, consider the set of temporal rule templates from Table 3 having as implicated clause $X_0(y_1 = start\_valley)$. To facilitate the notation, we denote with TR_1, TR_2 and TR_3 the three concerned templates, written in this order in the mentioned Table. Therefore $S_1 = \{TR_1, TR_2\}$, $S_2 =$

$\{TR_3\}$ and states used to calculate the entropy of the sets $A_j, j = 1..3$ are $\{s_1, \ldots, s_{100}, s_{300}, \ldots, s_{399}\}$. The encoding length for each subset $S \subseteq S_1 \cup S_2$ is calculated in the last column of the Table 5, value which is the sum of the templates encoding length (second column) and the exceptions encoding length (third column). As an observation, even if the set $\{TR_1, TR_2\}$ has more templates than the set $\{TR_3\}$, the encoding length for the two templates (14.34) is less than the encoding length of the last template (17.94). The conclusion to be drawn looking at the last column of the Table 5 is that the temporal meta rules, according to the MDLP criterion and inferred from the set $\{TR_1, TR_2, TR_3\}$ (based on the states $\{s_1, \ldots, s_{100}\}$, $\{s_{300}, \ldots, s_{399}\}$) is the subset $S = \{TR_1, TR_2\}$.

Table 5. The encoding length of different subsets of temporal rule templates having as implicated clause $X_0(y_1 = start\_valley)$, based on states $\{s_1, \ldots, s_{100}\}$ and $\{s_{300}, \ldots, s_{399}\}$

Subset S	Templates length	Exceptions length	Total length
$\{TR_1\}$	8.88	70.36	79.24
$\{TR_2\}$	7.48	66.64	74.12
$\{TR_3\}$	17.94	67.43	85.37
$\{TR_1, TR_2\}$	14.34	46.15	60.49
$\{TR_1, TR_3\}$	24.82	41.2	66.02
$\{TR_2, TR_3\}$	23.42	38.00	61.42
$\{TR_1, TR_2, TR_3\}$	31.72	30.43	62.15

Because the two definitions of temporal meta-rules differ not only in criterion (LCL, respectively MLDP), but also in the number of initial models (one, respectively at least two), the second inference process is applied in two steps. During the first step, temporal meta-rules are inferred from each set of temporal rule templates based on a single model. During the second step, temporal meta-rules are inferred from each set of temporal rules created during the step one and having the same implicated clause (see Fig. 4).

There is another reason to apply firstly the LCL criterion: the resulted temporal meta-rules are less redundant concerning the set of implication clauses and so the encoding procedures, used by MLDP criterion, don't need an adjustment against this effect.

6 Conclusions

In this article we constructed a theoretical framework for a methodology introduced in [9], which has as finality the discovery of knowledge, represented in the form of general Horn clauses, inferred from databases with a temporal

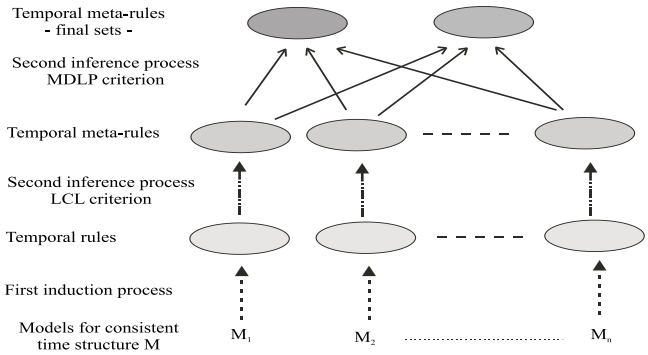

Fig. 4. Graphical representation of the second inference process

dimension. To obtain what we called *temporal rules*, a discretisation phase that extracts *events* from raw data is applied first, followed by an inductive phase, which constructs classification trees from these events. The discrete and continuous characteristics of an *event*, according to its definition, allow us to use statistical tools as well as techniques from artificial intelligence on the same data.

The theoretical framework we proposed, based on first-order temporal logic, permits to define the main notions (event, temporal rule, constraint) in a formal way. The concept of consistent linear time structure allows us to introduce the notions of *general interpretation*, of *support* and of *confidence*, the lasts two measure being the expression of the two similar concepts used in data mining.

Also included in the proposed framework, the process of inferring temporal meta-rules is related to a new approach in data mining, called *higher order mining*, i.e. mining from the results of previous mining runs. According to this approach, the rules generated by the first induction process are first order rules and those generated by the second inference process (i.e temporal meta-rules) are higher order rules. Our formalism do not impose which methodology must be used to discover first order rules. As long as these rules may be expressed according to the Definition 4, the strategy (here including algorithms, criterions, statistical methods), developed to infer temporal meta-rules may be applied.

It is important to mention that the condition of the existence of the limit, in the definition of consistent linear time structure, is a fundamental one: it express the fact that the structure M represents a homogenous model and therefore the conclusions (or inferences) based on a finite model $\tilde{M}$ for M are consistent. However, at this moment, we do not know methods which may certified that a given model is consistent. In our opinion, the only feasible approach to this problem is the development of methods and procedure for detecting the change points in the model and, in this direction, the analysis of the evolution of temporal meta-rules seems a very promising starting point.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[2] S. Al-Naemi. A theoretical framework for temporal knowledge discovery. In *Proceedings of International Workshop on Spatio-Temporal Databases*, pages 23–33, Spain, 1994.

[3] G. Berger and A. Tuzhilin. Discovering Unexpected Patterns in Temporal Data using Temporal Logic. *Lecture Notes in Computer Science*, 1399: 281–309, 1998.

[6] X. Chen and I. Petrounias. Discovering Temporal Association Rules: Algorithms, Language and System. In *Proceedings of the* 6^{th} *International Conference on Data Engineering*, page 306, San Diego, USA, 2000.

[5] X. Chen and I. Petrounias. A Framework for Temporal Data Mining. *Lecture Notes in Computer Science*, 1460:796–805, 1998.

[7] J. Chomicki and D. Toman. Temporal Logic in Information Systems. *BRICS Lecture Series*, LS-97-1:1–42, 1997.

[8] P. Cohen. Fluent Learning: Elucidating the Structure of Episodes. In *Advances in Intelligent Data Analysis*, pages 268–277. Springer Verlang, 2001.

[9] P. Cotofrei and K. Stoffel. Classification Rules + Time = Temporal Rules. In *Lecture Notes in Computer Science, vol 2329*, pages 572–581. Springer Verlang, 2002.

[10] P. Cotofrei and K. Stoffel. Rule Extraction from Time Series Databases using Classification Trees. In *Proceedings of IASTED International Conference*, pages 327–332, Insbruck, Austria, 2002.

[11] G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule Discovery from Time Series. In *Proceedings of the* 4^{th} *Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[12] E. A. Emerson. Temporal and Modal Logic. *Handbook of Theoretical Computer Science*, pages 995–1072, 1990.

[13] N. Friedman, K. Murphy, and S. Russel. Learning the structure of dynamic probabilistic networks. In *Proceedings of the* 14^{th} *Conference on Uncertainty in Artificial Intelligence*, pages 139–147. AAAI Press, 1998.

[14] G. Guimares. Temporal knowledge discovery for multivariate time series with enhanced self-organizing maps. In *Proceedings of the IEEE-INNS-ENNS Int. Joint Conference on Neural Networks*, pages 165–170. IEEE Computer Society, 2000.

[16] J. Han, W. Gong, and Y. Yin. Mining Segment-Wise Periodic Patterns in Time-Related Databases. In *Proceedings of the* 4^{th} *Conference on Knowledge Discovery and Data Mining*, pages 214–218, 1998.

[15] J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *Proceedings of International Conference on Data Engineering*, pages 106–115, Sydeny, Australia, 1999.

[17] F. Hoppner. Learning Temporal Rules from State Sequences. In *IJCAI Workshop on Learning from Temporal and Spatial Data*, pages 25–31, Seattle, USA, 2001.

[18] F. Hoppner. Discovery of core episodes from sequences. In *Pattern Detection and Discovery*, pages 199–213, 2002.

[19] P. Kam and A. W. Fu. Discovering Temporal Patterns for Interval-based Events. *Lecture Notes in Computer Science*, 1874:317–326, 2000.

[20] K. Karimi and H. Hamilton. Finding Temporal Relations: Causal Bayesian Networks vs. C4.5. In *Proceedings of the 12^{th} International Symposium on Methodologies for Intelligent Systems*, Charlotte, USA, 2000.

[22] E. Keogh and M. J. Pazzani. An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *Proceedings of the 4^{th} Conference on Knowledge Discovery and Data Mining*, pages 239–243, 1998.

[21] E. Keogh, S. Lonardi, and B. Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space. In *Proceedings of 8^{th} ACM SIKDD International Conference on Knowledge Discovery and Data Mining*, pages 550–556, Edmonton, Canada, 2002.

[24] E. J. Keogh and M. J. Pazzani. Scalling up Dynamic Type Warping to Massive Datasets. In *Proceedings of the 3^{rd} European Conference PKDD*, pages 1–11, 1999.

[23] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. Iterative Deepening Dynamic Time Warping for Time Series. In *Proceedings of Second SIAM International Conference on Data Mining*, 2002.

[25] W. Lin, M. A. Orgun, and G. J. Williams. Temporal Data Mining using Hidden Markov-Local Polynomial Models. In *Proceedings of the 5^{th} International Conference PAKDD, Lecture Notes in Computer Science*, volume 2035, pages 324 – 335, 2001.

[26] H. Loether and D. McTavish. *Descriptive and Inferential Statistics: An introduction.* Allyn and Bacon, 1993.

[27] W. Loh and Y. Shih. Split Selection Methods for Classification Trees. *Statistica Sinica*, 7:815–840, 1997.

[28] W. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83(403):715–725, September 1988.

[29] D. Malerba, F. Esposito, and F. Lisi. A logical framework for frequent pattern discovery in spatial data. In *Proceedings of 5th Conference Knowledge Discovery in Data*, 2001.

[30] S. Mangaranis. *Supervised Classification with Temporal Data.* PhD thesis, Computer Science Department, School of Engineering, Vanderbilt University, 1997.

[31] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *Proceedings of International Conference on Data Engineering*, pages 412–421, Orlando, USA, 1998.

[32] J. R. Quinlan. *C4.5: Programa for Machine Learning.* Morgan Kauffmann, San Mateo, California, 1993.

[33] J. R. Quinlan and R. L. Rivest. Inferring decision trees using Minimum Description Length Principle. *Information and Computation*, 3:227–248, 1989.

[34] J. Rissanen. Modelling by Shortest Data Description. *Automatica*, 14: 465–471, 1978.

[35] J. Rodriguez, C. Alonso, and H. Boström. Learning first order logic time series classifiers: Rules and boosting. In *Proceedings of 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 299–308, 2000.

[36] M. Spiliopoulou and J. Roddick. Higher order mining: Modelling and mining the results of knowledge discovery. In *Proceedings of the 2nd International Conference on Data Mining, Methods and Databases*, pages 309–320, UK, 2000.

[37] S. Tsumoto. Rule Discovery in Large Time-Series Medical Databases. In *Proceedings of the* 3^{rd} *Conference PKDD*, pages 23–31. Lecture Notes in Computer Science, 1074, 1999.