Classification of Multivariate Time Series via Temporal Abstraction and Time-Intervals Mining

Robert Moskovitch[1,2,3], Yuval Shahar[1]

[1] Department of Information Systems Engineering, Ben Gurion University, Israel.
[2] Telekom Innovation Laboratories at Ben Gurion University, Ben Gurion University, Israel.
[3] Department of Biomedical Informatics, Systems Biology, and Medicine,
Columbia University, New York, USA.
{robertmo,yshahar}bgu.ac.il

Abstract. Classification of multivariate time series data, often including both time points and intervals at variable frequencies, is a challenging task. We introduce a framework for classification of multivariate time series analysis, which implements three phases: (1) application of a temporal-abstraction process that transforms a series of raw time-stamped data points into a series of symbolic time intervals; (2) mining these intervals to discover frequent temporal patterns, using Allen's 13 temporal relations; (3) using the patterns as features to induce a classifier. Researchers can use different sets of temporal relations, and can vary an epsilon factor that represents flexibility in the nature of the temporal relations. We evaluated the framework on datasets in the domains of diabetes, intensive care, and infectious hepatitis, assessing the effects of the various settings of the KLS framework. Discretization using SAX led to better performance than using the Equal-Width method; knowledge-based abstraction, when available, was superior to both. Using three abstract temporal relations was superior to using the seven core temporal relations. Using an epsilon larger than zero tended to result in a slightly better accuracy when using SAX, but in a reduced accuracy when using EWD, and does not seem indicated. No feature selection method we tried proved useful. Regarding feature (pattern) representation, Mean Duration performed better than Horizontal Support (within the same entity), which performed better than the Binary (existence) representation method. Random Forest outperformed other induction methods we tried.

Keywords: Temporal Knowledge Discovery, Temporal Abstraction, Time Intervals Mining, Frequent Pattern Mining, classification.

1 Introduction

The increasing availability of time-stamped electronic data (e.g., Electronic Health Records, in the case of the medical domain) presents a significant opportunity to discover new knowledge from multivariate, time-oriented data, by using various data mining methods. Moreover, it enables researchers to perform classification and prediction tasks, based on the temporal data. However, temporal data include not only time-stamped raw data, or time *points* (e.g., white blood-cell count value of 4300, at 11:32am, on February 9th, 1991)), but also temporal *intervals*, possibly at a higher

level of abstraction, which are either a part of the original raw input data (e.g., administration of a medication for 4 days), or are *abstractions*, or interpretations, derived from them (e.g., three weeks of *severe anemia* or of *Bone-marrow toxicity Grade III*). Figure 1 illustrates the problem of having various time point series data and time interval series data representing various temporal variables in the same input dataset. Not only are the time points and intervals intermixed, but the time point series might be sampled or recorded at different frequencies: Sampling might occur at a fixed frequency, which may further vary for each type of variable, as shown in figure 1 for time series (a) and (b), which is often the case for automated sampling; or at random periods, as often occurs in manual measurements, as illustrated by time series (c). Often, certain time-stamped data points might be missing, or their measurements might include an error. Raw data (and certainly abstractions derived from the data) might also be represented by time intervals, such as medication-administration periods, as shown in series (d), in which the duration of the events is constant, and in series (e), in which the temporal duration is varying.

Designing algorithms capable of learning from such data, characterized in various forms, is a challenging topic in temporal data mining research, especially for the purpose of classification. Common methods for classification of multivariate time series, such as Hidden Markov Models (Rabiner, 1989) or recurrent neural networks, time series similarity measures (e.g., Euclidean distance or Dynamic Time Warping (Ratanamahatana and Keogh, 2002)), and time series feature extraction methods (e.g., discrete Fourier transform, discrete wavelet transform or singular value decomposition), cannot be directly applied to such temporal data.

Hu et al. (2013) point out that most of the studies in time series classification had unrealistic underlying assumptions, referring specifically to two relevant assumptions: (1) that perfectly aligned atomic patterns can be obtained, and (2) that the patterns are of equal lengths. While they are referring in their examples to uni-variate time series, the critique is at least as relevant to multivariate time series classification. However, the approach we will present here doesn't make any of these assumptions, and mines the data as they are, without any alignment pre-processing. Moreover, as we describe later, the temporal patterns we discover have various durations (lengths). Finally, the duration is not part of a pattern's explicit or implicit definition; the mined supporting instances can vary in their duration (length), but they are similar with respect to the temporal relations among their symbolic interval-based components.

Thus, in order to classify multivariate time series datasets having various forms of time stamped data, we propose to transform the time point series into symbolic time interval series representation. Such representation provides a uniform format of the various temporal variables, which enables to analyze the relations among the variables, such as by discovery of frequent temporal patterns. As we explain in Section 2, there are several approaches to that task, typically at a higher level of conceptual abstraction, which we refer to as temporal abstraction; some exploit context-sensitive knowledge acquired from human experts (Shahar et al, 1999), others are purely automatic (Azulay, 2007; Hoppner, 2001; Moerchen and Ultsch, 2005). Note that temporal abstraction can be applied to uni-variate as well as to multi-variate time series. In the case of multivariate series, each uni-variate time series is abstracted independently.

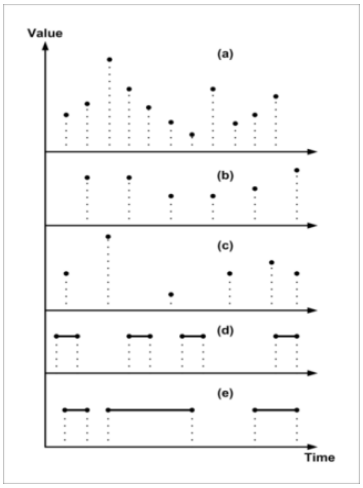

Fig 1. Time-point (a, b, c) and time-interval data (d, e).

Temporal abstraction enables us to overcome much of the problems of varying-frequency measurements and recordings, and of minor measurement errors and deviations in the raw data, through the creation of concepts that are no longer time-stamped, raw data, but rather interval-based *abstractions*, or interpretations, of these data, and through the smoothing effect of these abstractions. In many instances, the temporal-abstraction process also alleviates the problem of missing values, through a process of *interpolation* across temporal gaps that is inherent in several of the temporal-abstraction methods (see Section 2). Note that raw-data interval-based concepts such as "10 Days of Penicilin administration" might simply remain at the same level of abstraction. Whatever the temporal abstraction approach used, having the dataset represented by time intervals series, enables to discover frequent *Time Intervals Related Patterns* (*TIRPs*) (Kam and Fu, 2000; Hoppner, 2001; Moerchen, 2006; Sacchi et al, 2007; Patel et al, 2008; Papapetrou et al, 2009; Moskovitch and Shahar, 2013).

The use of TIRPs as features for the classification of multivariate time series was proposed first in (Moskovitch et al, 2009). This approach is inspired by the Bag-Of-Words in text categorization (Shahar, 1998), in which words are used as features for the classification of a given document. In our case, the discovered TIRPs from a given period of multivariate time series are used as features, which we call a Bag-Of-TIRPs, and is the focus of this paper. Since then several more studies were published exploring the use of frequent TIRPs as classification features for multivariate time series classification (Batal, 2012; Patel, 2008; Sacchi et al, 2007).

The *KarmaLegoSification* (KLS) framework that is at the focus of this paper is presented in the general block diagram shown in figure 2. The input data include multiple instances of entities (e.g., patients, or hardware devices) whose multiple variables (e.g., Hemoglobin value or Number of processes) are described by multivariate time-point series. The time-point series are abstracted, based on domain knowledge (a *knowledge-based* [KB] method), or on other computational means, and transformed into symbolic-value time intervals (e.g., *Moderate-Anemia* from t_1 to t_2).

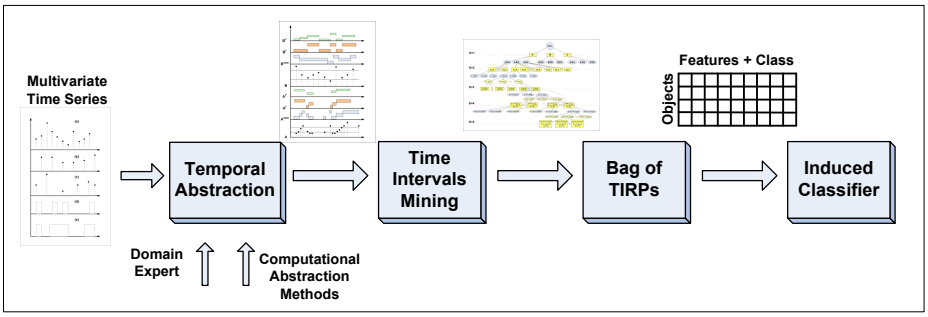

Fig 2. KarmaLegoSification - the overall temporal classification process. The raw-data time-point and time-interval series are abstracted into a uniform format of symbolic time intervals using domain knowledge or computational means. The symbolic time intervals are then mined and a tree of enumerated temporal patterns is generated. The frequent TIRPs are used as features by one or more classification algorithms to produce (induce) a classifier.

Then, the symbolic time intervals are mined using the *KarmaLego* algorithm, which we introduce in Section 3 in detail, to discover frequently repeating temporal patterns. The discovered TIRPs are used as features for the classifier.

After a TIRP tree is discovered using KarmaLego, several major application categories exist. These potential applications include: *Further temporal knowledge discovery*, in which a domain expert manually reviews the discovered TIRPs using a tool that we refer to as *KarmaLegoVisualization* (KLV) (Moskovitch and Shahar, 2009); *Temporal clustering*, in which each temporal pattern is considered as a cluster of entities (e.g., patients, mobile devices) who have similar temporal behavior; extraction of *Prediction* rules, based on the discovered TIRPs and the transition probabilities between the components of the patterns; and *Classification*, in which the discovered TIRPs are used as features for a classification task, which is the focus of the current paper.

The main contributions of the current paper can be summed up as follows:

1. *Introducing a comprehensive architecture and process, KarmaLegoSification* (KLS), *for classification of multivariate time series,* which is based on a temporal-abstraction process that can exploit domain knowledge, or, when needed, perform on-the-fly discretization, and which introduces several fundamental concepts that define the output of the time intervals mining process;
2. Evaluating the effect of applying *several types of temporal-abstraction methods, while also varying the number of bins*, and quantifying the effect of the abstraction method and of the number of bins on the eventual classification performance.
3. Introducing and analyzing *SingleKarmaLego* – a new algorithm for *fast TIRPs mining within a single entity that is represented by a set of symbolic time intervals,* given a set of TIRPs, for the purpose of classification of the entity, using these TIRPs as features. SingleKarmaLego is shown to be significantly superior, as the number of symbolic concepts in the domain and the length of each feature pattern increase, to a naïve method that attempts to query the single entity's record separately for each TIRP given as a feature to look for.

4. *Quantitatively evaluating the difference* in classification performance when using Allen's original *seven temporal relations,* versus using an abstract version o*f only three temporal relations*.

5. *Introducing two novel representation methods for features (i.e., TIRPs), which exploit the ex*istence of multiple instances within each time-oriented record, in addition to a Boolean r*epresentation, for the* purpose of classifica*tion*.

6. *Rigorously evaluating the KLS process and its several aspects, within several different real-life datasets*. As far as we know, no previous study has rigorously evaluated the effect all of these factors on the resulting classification, making it hard to assess their importance. Our evaluation addresses this situation.

The rest of this paper is organized as follows: We start by introducing briefly, in the Background (Section 2), the concepts of temporal data mining, temporal abstraction, temporal discretization, temporal interval-based mining, and classification based on patterns – specifically based on time intervals patterns. We then briefly introduce in the Methods (Section 3) a fast, time-interval mining method that we had developed, called KarmaLego, and show how it can be used to discover frequent TIRPs. We then explain exactly how TIRPs are used as features in our temporal data mining framework, and describe in section 4 our detailed evaluation of the efficacy of TIRP based classification of time oriented data. We summarize the main contributions and discuss their implications in Section 5.

2 Background

2.1 Temporal Data Mining

Temporal data mining is a sub-field of data mining, in which various techniques are applied to time-oriented data to discover *temporal knowledge*, i.e. knowledge about relationships amongst different raw-data and abstract concepts, in which the temporal dimension is treated explicitly. Unlike common data mining methods, which are static, often ignoring the temporal dimension, or using only concise statistical abstractions of it, temporal knowledge discovery presents significant computational and methodological challenges. However, temporal data mining offers considerable understanding of various scientific phenomena and the potential for creation of richer and accurate classification models, representing explicitly processes developing a long time. An excellent survey of temporal data mining can be found in (Roddick and Spiliopoulou, 2002).

2.2 Temporal Abstraction

Temporal abstraction (*TA*) is the segmentation and/or aggregation of a series of *raw*, *time-stamped*, multivariate data into a *symbolic time-interval* series representation, often at a higher level of *abstraction* (e.g., instead of a series of Hemoglobin or liver-function measurements, characterizations such as "3 weeks of moderate anemia", or "5 months of decreasing liver functions"), suitable for human inspection or for data mining.

TA, which typically includes also some form of interpolation (Shahar, 1999), solves

several common problems in mining raw time-series data, such as high variability in the sampling frequency and temporal granularity, minor measurement errors, and missing values, through the smoothing effect of the output abstractions. Thus, discovering frequent temporal patterns in multivariate temporal data can benefit from a preprocessing phase of converting the raw time-stamped data into a series of uniform, interval-based symbolic concepts. Figure 3 shows a TA process for one concept.

There are several approaches to the TA task; some exploit context-sensitive knowledge acquired from human experts, a method known as *knowledge-based temporal abstraction* (KBTA) (Salton et al, 1975); others are purely automatic, and rely mostly on a discretization of the raw values and concatenation (Azulay et al, 2007; Hoppner, 2002; Moerchen and Ultsch, 2005). *Temporal discretization* refers to the process of discretization of a time-series values, usually performed through unsupervised means, as a preprocessing step in transforming the time-stamped, raw-concept series into a set of symbolic, state-based time intervals.

2.3 Temporal Discretization

Although the knowledge-based TA approach is (Salton et al, 1975) very useful for the discovery of meaningful patterns, based on a domain knowledge base, especially in intensive-knowledge domains such as medicine, it might be less effective, as explained above, for other tasks, such as classification, clustering, and prediction. Moreover, unlike gradient and rate abstractions, which can often be computed and defined mathematically, state-abstraction definitions are not always provided by a domain expert, either because a state abstraction of the variable is not often used in that domain, or because there is no standard discretization for the variable.

In the light of this difficulty, we can consider the option of an automated analysis of longitudinal records and the discovery of knowledge within them through an automated process of discretization of the concepts' values; indeed, it is the default option that we explore in the current study when insufficient domain knowledge exists, especially in nonmedical domains.

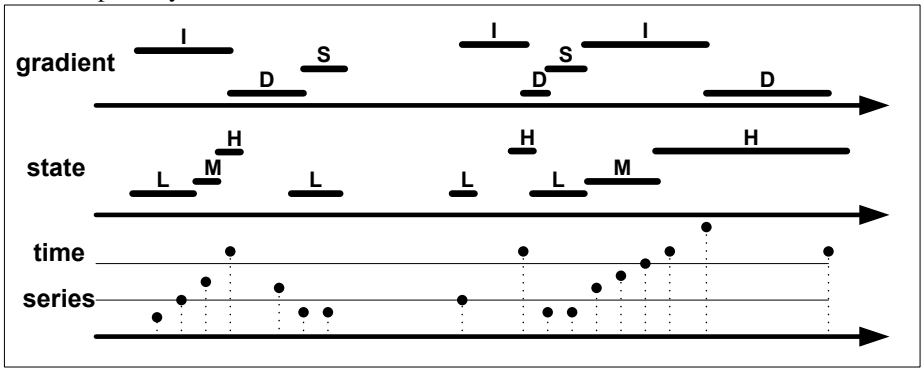

Figure 3: A series of raw time-series (at the bottom) is abstracted into an interval-based *state* abstraction that has three discrete values: Low (L), Medium (M), and High (H) (in the middle); and into a *gradient* abstraction (first derivative's sign) that has the values: Increasing (I), Decreasing (D), and Stable (S) (at the top).

Temporal discretization refers to the process of discretization of a time-series values, usually performed through unsupervised means, as a preprocessing step in transforming the time-stamped, raw-concept series into a set of symbolic, state-based time intervals. Temporal Abstraction for time series mining in the form of time intervals was already proposed by Hoeppner (2002). Several common discretization methods, such as *Equal Width Discretization (EWD),* which uniformly divides the ranges of each value, and *Equal Frequency Discretization* (*EFD*), do not consider the temporal order of the values; other methods, such as *Symbolic Aggregate approXimation* (*SAX*) (Lin et al, 2003) (which focuses on a statistical discretization of the values) and *Persist* (Moerchen and Ultsch, 2005) (which maximizes the duration of the resulting time intervals), explicitly consider the temporal dimension. In previous studies, we have compared several versions of these methods, especially for the purpose of discretization of time-oriented clinical data (Azulay et al, 2007) and eventually for the purpose of classification (Moskovitch and Shahar, 2009).

In the evaluations performed in the current study, we have used interval-based abstract concepts generated through the use of both knowledge-based temporal abstraction and automated temporal discretization using equal-width discretization and SAX (Lin et al, 2003).

2.5 Mining Time-Intervals

Mining time-intervals is a relatively young research field that has mostly sprung during the past decade. Most of the methods use some subset of Allen's temporal relations (Villafane et al, 2000). As was presented by Moerchen (2006) these suffer from "crispiness" and being sometimes over-detailed for knowledge discovery purposes, thus, in our framework two ways of overcoming this are proposed and rigorously evaluated. One of the earliest studies in the area is that of Villafane et al. (2000), which searches for containments of time intervals in a multivariate symbolic time interval series. Kam and Fu (2000) were the first to use all of Allen's temporal relations to compose time interval series, but their patterns were ambiguous, since the temporal relations among the components of a pattern are undefined, except for the temporal relations among all of the pairs of successive intervals.

Höppner (2001) was the first to define a non-ambiguous representation of time-interval patterns that are based on Allen's relations, by a k^2 matrix, to represent all of the pair-wise relations within a k-intervals pattern. In the rest of this paper, we shall refer to a conjunction of temporal relations between pairs of intervals as a *Time Interval Related Pattern* (*TIRP*). The formal definition of a TIRP appears in Section 3 (Definition 5). Papapetrou et al. (2009) propose a hybrid approach H-DFS, which combines the first indexing the pairs of time intervals and then mining the extended TIRPs in a candidate generation fashion. Papapetrou et al. used only five temporal relations: meets, matches (equal, in terms of Allen's relations), overlaps, contains, and follows, similar to Allen's temporal relations, and introduced an *epsilon threshold*, to make the temporal relations more flexible.

ARMADA, by Winarko and Roddick (2007), is a relatively recent projection-based efficient time intervals mining algorithm that uses a candidate generation and mining iterative approach. Wu et al. (2007) proposed TPrefixSpan, which is a modification of

the PrefixSpan sequential mining algorithm (Pei et al, 2001) for mining non-ambiguous temporal patterns from time interval based events. Patel (Patel et al, 2008) introduced IEMiner - a method inspired by Papapetrou's method, which extends the patterns directly, unlike Papapetrou et al.'s method, as in fact is done in the KarmaLego method (see Section 3.6). Patel et al. (2008) had compared their method runtime to TPrefixSpan (Wu et al, 2007) and H-DFS (Papapetrou et al, 2009) and found their method to be faster. Moskovitch and Shahar introduced the KarmaLego framework (Moskovitch and Shahar, 2009; Moskovitch and Shahar, 2013), which extends TIRPs directly and exploits the transitivity property to generate candidates efficiently, which was found faster than H-DFS, IEMiner and ARMADA. (We present the KarmaLego algorithm briefly in Section 3).

Other methods for time intervals mining were proposed, which either do not use Allen's temporal relations (Moerchen, 2006), or use only a subset of these relations, abstracted into a super-relation, such as *Precedes*, which is the disjunction of *Before*, *Meets*, *Overlaps*, *Equal*, *Starts*, and *Finished-By*, and which can form a basis for the discovery of a set of temporal association rules (Sacchi et al, 2007).

2.5 Classification via Frequent Patterns

The increased attention to the subject of mining time intervals has led several research groups to simultaneously propose using the discovered temporal patterns for classifying multivariate time series (Batal et al, 2012; Patel et al, 2008), including the suggestion of a highly preliminary version of the KarmaLegoS framework (Moskovitch et al, 2009) whose full details and rigorous evaluation we present in Sections 3 and 4, respectively.

Patel el. at (2008) presented a time intervals mining algorithm, called IEMiner and used the discovered patterns for classifying multivariate time series. A patterns selection measure, GAIN, which is actually a feature selection method, is proposed. The GAIN measure is an entropy-based measure, assumes that temporal patterns that occur in only one class are more discriminative.

In addition to GAIN the IEClassifier is introduced, a classification method that is designed specifically to classify data using temporal patterns (Patel et al, 2009). Two versions of IEClassifier, each with a different criterion, were proposed. In the Best_Confidence version, the class having the highest confidence is selected, while in the Majority_Class version, the class to which the majority of the patterns discovered belong is selected. The two versions of IEClassifier were compared with common classifiers, such as C4.5 and SVM, when applied to a non-temporal representation of the data. In the data used in the study, The Majority_Class out-performed the other classification methods. Batal et al. (2012) present a study in which time intervals patterns classify multivariate time series, using an a priori approach, STF-Mine, for the discovery of temporal patterns, and the X^2 (chi-square) measure for pattern selections.
Batal et al (2012) presented an approach for targeted events prediction based on time intervals patterns events. For that, they used only two temporal relations: "Before" (which is similar in semantics to Allen's relation) and "Overlaps" (the later relation is

actually a disjunction of all Allen's temporal relations, except "Before"). Moerchen and Fradkin (2010) introduce the use of semi-intervals mining with partial order to enable more flexibility, as opposed to Allen's (1983) interval-based temporal relations. Two flexible representations were proposed, termed SISP and SIPO, for semi-intervals patterns, inspired by Wu and Chen (2007). The authors evaluate their pattern representation, comparing it to Allen's temporal relations, on seven datasets, some originating from time intervals data, and some abstracted from multivariate time series using Persist (Moerchen and Ultsch, 2005), using the Precision and Recall measures to assess directly the value of the new features. The authors' results show that an additional number of frequent patterns were found with their disjunctive representation. However, since training and classification were not used, and accuracy results were not reported, it is hard to assess the effectiveness of these features, unlike the experiments that we present in our current study. Although in recent years several studies have started to use time interval patterns for classification purposes, no study investigated the influence of the abstraction method on the accuracy, as well as the number of bins, which is one of the several contributions of our work.

3. Methods

We start by introducing our method for discovering frequent patterns, the KarmaLego framework; we then explain in detail how we extended that framework into the KarmaLegoS classification method, and in the next section, how we rigorously evaluated that method in several different domains.

3.1 KarmaLego – Fast Time-Intervals Mining

The discovery of *Time Interval Related Patterns* (*TIRPs*) is computationally highly demanding, since it requires generating all of Allen's seven basic temporal relations. For example, a naive generation of all TIRPs having 5 symbolic time intervals, with all possible temporal relations among them specified unambiguously, requires in theory generating up to $7^{((5^2 - 5)/2)} = 7^{10} = 282{,}475{,}249$ candidate TIRPs. In general, given a TIRP having k time intervals, we will have up to $7^{((k^2 - k)/2)}$ candidate TIRPs.

To overcome this difficulty, we have developed *KarmaLego*, a fast algorithm which generates all of the patterns efficiently by extending TIRPs directly, and exploiting the transitivity property of the temporal relations to remove unrealistic candidates.

To increase the robustness of the discovered temporal knowledge, KarmaLego uses a flexible version of Allen's seven temporal relations. This is achieved by adding an epsilon value to all seven relations, as explained in the next subsection. Furthermore, we also limit the *before* temporal relation by a maximal allowed gap, as proposed by Winarko and Roddick (2007).

To define formally the problem of mining time intervals and relevant measures for the classification task, we first present several basic definitions. These definitions will be used in the description of the methods.

3.2 Definitions

To better understand the KarmaLegoS methodology, we introduce several key definitions, several of which extend our initial definitions for the KarmaLego framework (Moskovitch and Shahar, 2013).
To define a flexible framework of Allen's temporal relations in KarmaLego, two relations are defined on time-stamped (point-based) data, given an epsilon value.

Definition 1. Given two time-points t_1 and t_2:

$t_1 =^{\varepsilon} t_2$ *iff* $|t_2 - t_1| <= \varepsilon$

and

$t_1 <^{\varepsilon} t_2$ *iff* $t_2 - t_1 > \varepsilon$.

Based on the two relations $=^{\varepsilon}$ and $<^{\varepsilon}$ and the epsilon value, a flexible version for all of Allen's seven relations is defined, extending Papapetrou's definition, as shown in figure 4.

The introduction of the epsilon parameter to Allen's full set of temporal relations maintains the *Jointly Exhaustive and Pairwise Disjoint* (*JEPD*) conditions, as will be shown soon. The Jointly Exhaustive condition comes from probability theory and means that a set of events is jointly exhaustive if at least one of the events must occur. In the context of temporal relations it means that the set of temporal relations, which are defined, must cover all of the optional temporal relations among two time intervals.

The Pairwise Disjoint condition means that two sets A and B are disjoint if their intersection is the empty set. In the context of temporal relations it means that the introduction of the epsilon value as defined in definition 1 and figure 4 keeps the set of the temporal relations mutually exclusive. This is indeed true, since the epsilon-extended temporal-relation definitions appearing in figure 4 imply that for any two time intervals, exactly one (epsilon-extended) temporal relation applies.

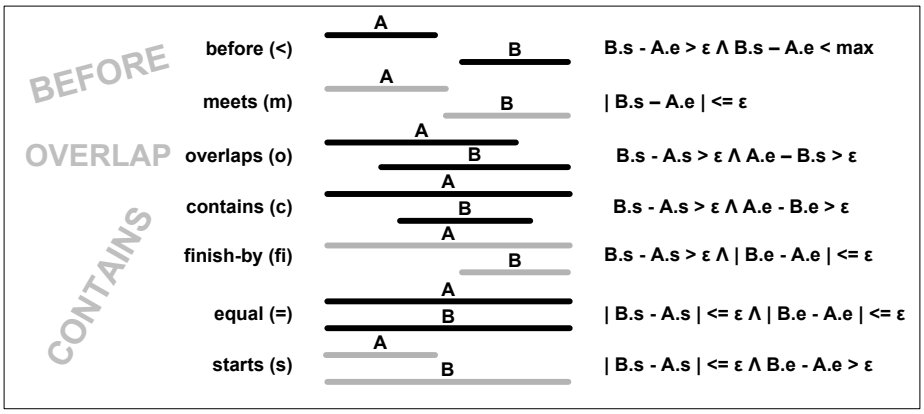

Figure 4. A flexible extension, using the same epsilon value, for all Allen's seven relations, using the same Epsilon for all the relations.

In addition to a flexible (epsilon-based) semantics for Allen's seven temporal relations, we introduce a set of three abstract temporal relations (shown in figure 4 in grey labels): BEFORE = {before | meet} that is the disjunction of Allen's before and meets, OVERLAP that is Allen's overlap and CONTAIN = {finish-by | contain | start-by | equal} that is the disjunction of finish-by, contain, start-by and equal.

Definition 2. A *symbolic time interval*, $I = <s, e, sym>$, is an ordered pair of time points, start-time (s) and end-time (e), and a symbol (sym) that represents one of the domain's symbolic concepts. When referring to these properties we will use the following $I.s$ for its start-time, $I.e$ for its end-time, and $I.sym$ for its symbol.

Definition 3. A *symbolic time interval series*, $IS = \{I^1, I^2, .. , I^n\}$, where each I^i is a symbolic time interval, represents a series of symbolic time intervals, over each of which holds a start-time, end-time and a symbol.

Definition 4. A *lexicographic symbolic time-interval series* is a symbolic time interval series, sorted in the lexicographical order of the start-time, end-time using the relations $<^{\varepsilon}, =^{\varepsilon}$ and the symbols, $IS = \{I^1, I^2, .. , I^n\}$, such that:

$$\forall I^i, I^j \in IS (i < j) \wedge ((I^i_{.s} <^{\varepsilon} I^j_{.s}) \vee (I^i_{.s} =^{\varepsilon} I^j_{.s} \wedge I^i_{.e} <^{\varepsilon} I^j_{.e}) \vee (I^i_{.s} =^{\varepsilon} I^j_{.s} \wedge I^i_{.e} =^{\varepsilon} I^j_{.e} \wedge I^i_{.sym} < I^j_{.sym}))$$

Since in our problem definition the time intervals are ordered lexicographically, we use only the seven temporal relations shown in figure 5.

Definition 5. A non-ambiguous lexicographic *Time Intervals Relations Pattern* (*TIRP*) P is defined as $P = \{I, R\}$, where $I = \{I^1, I^2, .., I^k\}$ is a set of k symbolic time intervals ordered lexicographically and

$$R = \bigcap_{i=1}^{k-1} \bigcap_{j=i+1}^{k} r(I^i, I^j) = \{r_{1,2}(I^1, I^2), .., r_{1,k}(I^1, I^k), r_{2,3}(I^2, I^3), .., r_{k-1,k}(I^{k-1}, I^k)\}$$

defines all the temporal relations among each of the $(k^2-k)/2$ pairs of symbolic time intervals in I.

Figure 5 presents a typical TIRP, represented as a half-matrix of temporal relations. We will usually assume such a representation through the description of the KarmaLego algorithm.

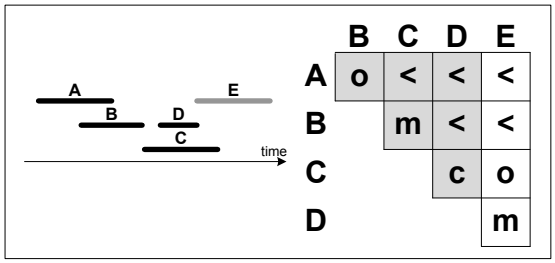

Figure 5. An example of a Time-Interval Related Pattern (TIRP), represented by a sequence of five lexicographically ordered symbolic time intervals and all of their pair-wise temporal relations. On the left, the actual five-symbols TIRP is displayed graphically, while on the right, a half matrix representation is given presenting the pairwise temporal relations between each two symbolic time intervals. Interval E is a candidate symbol that is being added to the current

TIRP, and its relations with the other four symbolic intervals are shown in the last column of the half matrix.

One potential problem with Definition 5 is that it is purely qualitative; it ignores the precise quantitative durations of the time intervals that are the components of the TIRP. We focus on this problem, and on a possible solution of it, in Section 3.7, when we propose to pre-cluster the time intervals by duration. (In the Discussion, we also consider the other potentially quantitative aspect, i.e. the nature of the temporal relationship itself, such as the duration of the gap in the case of the *before* relation.)

Figure 5 presents the output of a temporal interval mining process, i.e., an example of a TIRP, represented, for efficiency purposes, as a half matrix. Thus, the half matrix on the right part of figure 5 presents all of the pair-wise temporal relations among the TIRP's symbolic time intervals, ordered lexicographically, that defining it in a canonical, non-ambiguous fashion. Note that *half-matrix* representation (as opposed to a full matrix) is possible due to the fact that each of Allen's temporal relations has an inverse; and that the *canonical* aspect is due to the lexicographic ordering, which leads to a unique half matrix for each TIRP.

Definition 6. Given a database of $|E|$ distinct entities (e.g., different patients), the *vertical support* of a TIRP P is denoted by the cardinality of the set E^P of distinct entities within which P holds at least once, divided by the total number of entities (e.g., patients) $|E|$: $ver\_sup(P) = |E^P| / |E|$. The vertical support is the term usually referred to as *support* in the context of association rules, itemsets, and sequential mining.

When a TIRP has vertical support above a minimal predefined threshold, it is referred to as *frequent*. Note that in pattern mining, such as association rules, sequential mining and time intervals mining, *support* typically refers to the percentage of entities in the database supporting a pattern, which is actually the vertical support presented in Definition 6.

Since a temporal pattern can be discovered multiple times within a single entity (e.g., the same TIRP appears several times (multiple instances) in the longitudinal record of the same patient), we distinguish between two types of support: the *vertical* support and the *horizontal* support, which represents the number of patterns discovered within the longitudinal record of a specific entity, as defined in definition 7.

Definition 7. The *horizontal support* of a TIRP P for an entity e_i (e.g., a single patient's record), $hor\_sup(P, e_i)$ is the number of instances of the TIRP P found in e_i. For example, the number of times that a particular temporal pattern was found in a particular patient's record.

Definition 8. The *mean horizontal support* of a TIRP P is the average horizontal support of all the entities E^P supporting P (i.e., for all entities that have a horizontal support ≥ 1 for TIRP P).

$$MeanHorizontalSupport(P, E^P) = \frac{\sum_{i=1}^{|E^P|} hor\_\sup(P, e_i)}{|E^P|}$$

Definition 9. The mean duration of the *n* supporting instances of the same *k-sized* TIRP *P* within an entity *e* (e.g., within a single patient's record; note that, per Definitions 7 and 8, an entity may have several supporting instances of a TIRP) is defined by:

$$MeanDuration(P, e) = \frac{\sum_{i=1}^{n} \left(Max_{j=1}^{k} I_e^{i,j} - I_s^{i,1} \right)}{n}$$

where $I^{i,1}_s$ is the *start-time* (s) of the first time interval in the *i* instance (among *n* instances), and the Max operator selects the time interval having the latest *end-time* (e) among the *k* time intervals of an instance *i*. Note that according to the lexicographical order (Def 4) the first time interval must have the earliest start-time, while the latest end-time can be of any of the time intervals in the instance.

As we show Section 4, the horizontal support and the mean duration of TIRP are potentially useful metrics when using TIRPs as features, for classification purposes.

Definition 10. The time-intervals mining task:

Given a set of entities *E*, described by a symbolic time-interval series *IS*, and a minimum vertical support *min_ver_sup*, the goal of the tasks is to find all the TIRPs whose vertical support is above the *min vertical support* threshold.

3.3 The KarmaLego Algorithm

KarmaLego is a TIRP-discovery algorithm that we have defined and evaluated previously, as part of our research into effective TIRP-based classification (Moskovitch and Shahar, 2013). Here, we summarize the key features of KarmaLego that are relevant to the current study.

The main body of the KarmaLego algorithm consists of two main phases, Karma and Lego (algorithm 1). In the first phase, referred to as *Karma*, all of the frequent two-sized TIRPs, $r(I_1,I_2)$ having two symbolic time intervals I_1 and I_2 that are ordered lexicographically and are related by *r*, a temporal relation, are generated, discovered, and indexed, using a breadth-first-search approach. In the second phase, referred to as *Lego*, a recursive process extends the frequent 2-sized TIRPs, referred to as T^2, through efficient candidate generation, using the Lego procedure, into a tree of longer frequent TIRPs consisting of conjunctions of the 2-sized TIRPs that were discovered in the Karma phase. Eventually a tree of all frequent TIRPs is discovered and returned (Figure 6).

The data structure supporting this process is a hierarchy of frequent TIRPs, as illustrated in Figure 6. A TIRP is represented by a vector of symbols in their lexicographical order, and a two dimensional array for the temporal relations representation, corresponding to the illustration of the half matrix of temporal

relations shown in figure 4. Each TIRP is supported by a list of instances that are represented by a vector of time intervals instances that satisfy the TIRP's constraints regarding the symbols and temporal relations.

Algorithm 1 – KarmaLego

Input:

db – A database of |E| entities, representing for each entity *e* a set of symbolic time intervals out of |S| symbols;

min_ver_sup – the minimal vertical support threshold;

Output: *T* – an enumerated tree of all frequent TIRPs

1. T ← Karma(*db*, min_ver_sup)
2. Foreach $t \in T^2$ // T^2 is *T* at the 2nd level
3. Lego(T, t, min_ver_sup)
4. End Foreach
5. return T
6. End

Note that, for robustness purposes, we are using the flexible version of Allen's temporal relations (see Definition 1). However, *the KarmaLego algorithm is oblivious to the precise definition of temporal relations*. This is demonstrated in this study, in which two sets of temporal relations are used Allen's original seven temporal relations vs an abstract form of three temporal relations.

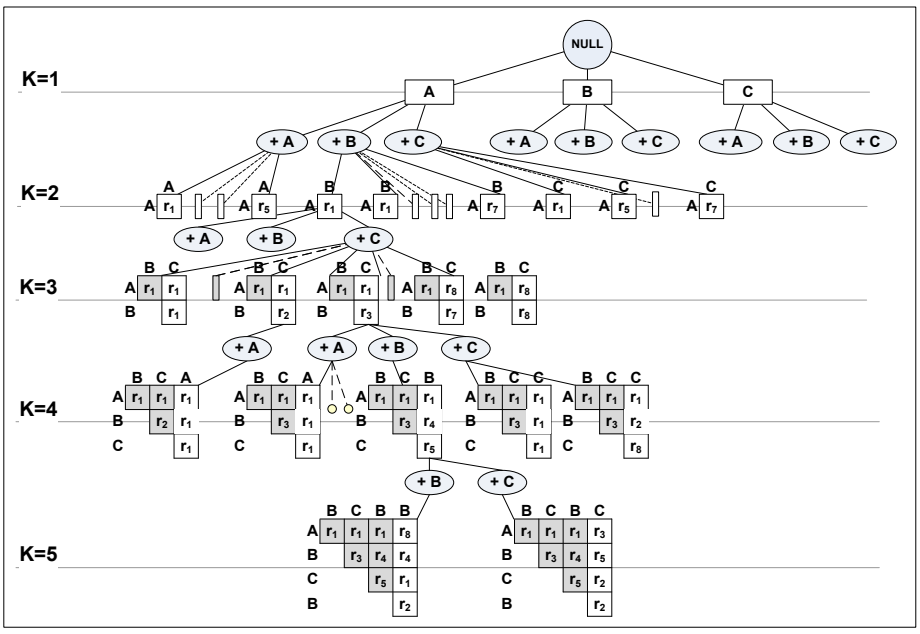

Figure 6. A KarmaLego enumeration tree, in which the direct expansion of TIRPs is performed. Each node represents a TIRP as a half matrix of its relevant temporal relations.

3.3.1 Karma

In algorithm 1 (*Karma*[1]), the first level of the enumeration tree is constructed based on the set of the frequent symbols *S*, as shown in figure 6. Then, in a single scan of the data each entity e's pairs of symbolic time intervals are indexed by their two symbols and the relation r among them, which creates the second level of the enumeration tree T^2. Finally, the 2-sized TIRPs that are not frequent are removed.

To enable later a highly efficient retrieval of the 2-sized TIRPs for two given symbols (i.e., pair-wise relations between intervals that are annotated by two given symbols) for the recursive Lego step, T^2 is implemented by a two-dimensional square array of size $|T^1|^2$. $|T^1|$ is the number of the frequent symbols found in Karma. Each cell in the two dimensional array contains a vector of size |R| (the number of temporal relations) that holds a HashMap, which contains all of instances of the indexed pairs. Thus, retrieval of the indexed pairs from T^2 in the Lego phase is performed in O(1), using as indices the two symbols and the temporal relation. As we explain later, this structure is also highly useful in SingleKarmaLego.

Algorithm 2 – Karma

Input:
db – A database of |E| entities (the overall set of entities being referred to as E), representing for each entity *e*, the lexicographically sorted vector of its symbolic time intervals, *e*.I;
min_ver_sup – the minimal vertical support threshold;
Output: *T* – an enumerated tree of up to 2-sized frequent TIRPs

1. $T \leftarrow \emptyset$
2. Foreach $e \in E$
3. Foreach $I^i, I^j \in e.I \wedge i<j$
4. $r \leftarrow$ the temporal relation among I^i, I^j
5. Index(T^2, $< e.I_{.sym}^{\ i}, r, e.I_{.sym}^{\ j} >$)
6. End Foreach
7. End Foreach
8. Foreach $t \in T^2$
9. if ver_sup(*t*) < min_ver_sup
10. Prune(t)
11. End Foreach
12. return *T*
13. End

[1] Karma – The law of *cause* and *effect* originated in ancient India and is central to Hindu and Buddhist philosophies.

3.3.2 Lego

The *Lego*[2] (Algorithm 3) receives a *k*-sized TIRP and extends it by all of the frequent symbols in T^1 and a set of predefined desired temporal relations *R* (Allen's seven relations, or other versions). This *direct extension* approach of generating TIRPs candidates is part of the KarmaLego framework's contribution, and is presented in more detail elsewhere (Moskovitch and Shahar, in Press). Then a new TIRP is created of the extended size and a symbol *s* and a temporal relation *r* are set to t^c and the method Generate Candidate TIRPs generates all the corresponding TRIPs based on the various temporal relations settings. Then, for each candidate *c* in *C*, if the frequency of the supporting instances is above the minimal vertical support threshold it is further extended by calling Lego.

The *candidate generation* process can be performed by using a naïve candidate generation method, as has been done in previous algorithms, or through a much faster method, exploiting the transitivity of temporal relations, which leads to significant pruning of unviable candidate TIRPs, which is implemented in KarmaLego (Moskovitch and Shahar, 2013).

Algorithm 2 – Lego(T, t, min_ver_sup)

Input:

T – the enumeration tree after Karma was ran,

t – a TIRP that has to be extended,

min_ver_sup - the minimal vertical support threshold

Output: void

1. Foreach $sym \in T^1$
2. Foreach $r \in R$
3. Create new t^c of size (t.size + 1)
4. t^c.s[t^c.size-1] ← *sym*
5. t^c.tr[t^c.tr_size-1] ← *r*
6. *C* ← 0
7. C ← Generate_Candidate_TIRPs(t^c, *1*)
8. Foreach $c \in C$ // candidates
9. Search supporting instances(*c*, T^2)
10. if(ver_sup(*c*) > min_ver_sup)
11. *T* ← *T* U *c* // c is frequent
12. Lego(*T*, *c*, min_ver_sup)
13. End Foreach
14. End Foreach
15. End Foreach
16. End

[2] Lego – A popular game, in which modular bricks are used to construct different objects.

The entire process of searching for supporting instances is done using the 2-sized TIRPs. In each extension the TIRP is extended by a relation to a symbol that has to be supported by the 2-sized TIRPs and its corresponding relations to the previous symbols. This process is repeated as necessary, while maintaining each time the correct entity and the relevant set of symbolic time intervals for which the (partial) conjunction holds to far. Finally, the vertical support (number of entities for which the full extended TIRP holds) is checked to determine whether to add it to the TIRP tree.

KarmaLego discovers all of the frequent TIRPs of an entity (e.g., a patient record), including all of the instances of that TIRP over time within that entity. This aspect actually enables us later to calculate the novel TIRP representations of horizontal support and mean duration.

However, note that there is no additional cost in discovering all of the instances, since finding all instances of a pattern for each entity is actually obligatory, in order to ensure completeness in the discovery of temporal patterns, even just for a binary (existence/non-existence of the TIRP in each entity) representation!

We are highlighting this important point, since in typical previous temporal-pattern discovery methods, the algorithm stops when discovering the first instance of a new pattern, and does not include the next one. We consider such an approach to be wrong, since in general, it is impossible to know ahead of time, which instance of the pattern, for a given entity, will be (or can be) extended later to support extended TIRPs. Thus, all instances of the TIRP must be found.

In other words, if one's objective (as is the case here, as well as when we later look at the TIRPs as features that might hold for each single entity) is completeness, i.e., to discover, within each entity, *all* of the different TIRPs existing within it, not just soundness, i.e., discovering *some* TIRPs that exist in it, then ALL of the k-sized TIRPs *must* be discovered, in order to enable discovery of all of the k+1 sized TIRPs (where k is the number of symbolic intervals in the TIRP), even within the same entity.

3.4 KarmaLegoSification - Classification of Multivariate Time Series via TIRPs

The KarmaLegoSification (KLS) framework for the classification of multivariate time series via TIRPs includes four main components, as was shown in figure 2: Temporal Abstraction, Time Intervals Mining, TIRP-based Feature Representation, and Classifier induction, thus producing a TIRP-based Classifier.

Each component in the KLS framework (see Figure 2) has several potential settings. The temporal-abstraction method settings include the temporal-abstraction or temporal-discretization method itself (e.g. SAX, EWD, knowledge-based) and the number of discrete states (symbols) generated. The time-intervals mining-method (i.e., KarmaLego) settings include the epsilon value and the minimal vertical threshold. The TIRP-based representation settings include the feature-selection method and the representation method of the TIRPs (a Boolean value representing an

existence of the TIRP, versus actual numerical horizontal support) and the classifier-induction setting is the type of induction method used.

Before we delve into the KLS framework, it is important to understand the difference between the process of TIRP-based classification and (cross) validation, and the one often used in the context of standard classification experiments.

3.4.1 Time Intervals Mining: The Training versus the Classification Phases

When using TIRPS for classification purposes, an important distinction must be made between the training phase and the classification (or validation) phase. This distinction usually does not exist in the case of a standard data mining task.

First, note that when classifying static data, such as determining whether a particular geological site contains oil, the classified entities are represented by a static features vector, such as the type of soil and its mineral content. However, in the case of temporal classification, the entities are represented by multivariate time series, such as (in the case of clinical assessment) longitudinal, co-occurring, multiple laboratory tests, time intervals of various known diagnoses, or time intervals of various treatments, all accumulating over time. Thus, the features mostly consist of the discovered TIRPs.

Second, note as a result of the data representation nature, and the semantics of the TIRP discovery process, the features vector (consisting mostly of TIRPs), representing any particular entity, may vary according to the specific (discovered) TIRP tree that was used to represent the multivariate time series of that entity; the contents of that TIRP tree, in turn, depend on the particular entities used to generate the tree, since within each subset of entities used for the generation process, different TIRPs might pass the frequency threshold.

This insight has three major implications: (1) to the *training phase*, since *the features (TIRPs) may be different for each training set* - thus, when new entities are added to a time-oriented training set, the mining process should be applied again, using *all* of the entities, to produce the correct features (TIRPs), unlike the case of static classification tasks, in which the basic features (e.g., mineral content) are always the same, and are not dependent on the rest of the entities in the same class; (2) to the *classification phase*, since the TIRPs, once discovered using a population of entities, have to be detected (i.e., determined as existing or not) within the (longitudinal) record of a single entity, to assign, for that entity, a value to each feature; and (3) to the *evaluation phase*, since to perform the popular process of cross validation, in which a different portion of the dataset is used each time as a training set and the other portion is used as a testing set, the mining process that defines *the very features to be used* (even before the processes of feature selection and classifier induction take place) should be repeated for the particular training set used, *in each iteration*.

Thus, unlike the case of static data-mining tasks, in which we can use a static matrix of entities versus features, and, during the cross validation process, select each time a different subset of the feature-matrix rows (each row standing for an entity, represented by a vector of values of the *same* set of features), here *the cross-*

validation process must first repeat the mining process for the selected subset of entities, extract the features (TIRPs) specific to that subset, and construct a new feature matrix for each iteration in the cross validation, *before* the feature-selection and learning (induction) phases.
Note that this dynamic state of affairs is qualitatively not unlike the case of text mining, in which, to construct an appropriate vector-space model, the cross validation process might involve re-computing a list of relevant terms, using measures such as term frequencies and inverse (within) document (term) frequencies (TF/IDF); but in the case of temporal classification, we encounter the issue at a significantly higher level of complexity. Note also that this observation is of course relevant to any pattern-based method for classification of multivariate time series, or, indeed, to any other data mining/machine learning task having dynamic features, that is, features that are not predefined, but rather, are defined on the fly upon examination of the training set.

3.4.2 The KarmaLegoSification Methodology

To classify an entity, its symbolic time intervals series must be searched for the features, i.e., frequent TIRPs. When mining a single entity, i.e., finding all of the [frequent] TIRPs for a particular subject (e.g., a patient), the notion of minimal vertical support is meaningless, and thus, all of the relevant TIRPs existing within the record of that entity should be discovered. Moreover, in general, not *all* of the discovered TIRPs are relevant for the classification task: in reality, only the TIRPs that appear in the training data, and were thus defined as features, should be searched, as shown in Algorithm 4. Thus, we modified KarmaLego to be applied to a single entity, a process that we refer to as SingleKarmaLego.

However, before describing the process of mining a single entity, we will briefly describe the process of training. Given a set of E entities divided into C classes, each class is abstracted temporally using the same definitions, and mined for TIRPs, using the same minimal vertical support threshold. The result is C sets of TIRPs – each of which is related to one class. Finally, the union of all of the discovered TIRPs from each class is used for the feature-matrix representation. This set of TIRPs may include TIRPs that appear in all of the classes, or in some of them. Although the set of TIRPs can be reduced by performing feature selection, eventually there is a set of TIRPs P, which can be used for the training and later for classification. Note that we do not mine all of the set E (as a single set) for TIRPs, since certain TIRPs characterizing some of the classes might not necessarily pass the frequency threshold when diluted by entities from all other classes.

Thus, in the classification phase, each single entity e has to be mined separately to search for occurrences of the specific TIRPs in P, as will be shown in Algorithm 4 – SingleKarmaLego. Note that since the vertical support is meaningless, there is no stopping criterion for the algorithm and any discovered TIRP will be "frequent". However, since we are interested in discovering only the TIRPs that were discovered in the training phase, the algorithm should search for these TIRPs only. In Algorithm 4, the input is a set of TIRPs P, which was discovered in the training set (the unification of all the classes), which is actually the enumeration tree that was

discovered. Thus, the goal is to follow the enumeration tree which is represented by P in order to generate and discover all the existing TIRPs in the entity. The mining process is actually the same as was described in KarmaLego, but without considering the minimal vertical support.

Algorithm 4 – SingleKarmaLego

Input:

entity_sti – the single entity's symbolic time intervals records; that is, e.I

T – the enumeration tree of the [full] training set

Output: Single_*T* – an enumerated tree of TIRPs

1. Single_*T* ← SingleKarma(entity_sti, *T*, epsilon)
2. Foreach $t \in$ Single_T^2 // *Single_*T^2 is Single_*T* at the 2nd level
3. SingleLego(Single_*T*, *T*, t, 2, epsilon)
4. End Foreach
5. return Single_*T*
6. End

Algorithm 5 is similar to Algorithm 2 (Karma), but instead of generating all the possible TIRPs, it searches only for the TIRPs at the first and the second level in the given enumeration tree T that was discovered in the training set phase. After that, for all the TIRPs at the third level that appear in T, the SingleLego algorithm is called to further generate and search the extended TIRPs of this path in the tree T.

Algorithm 5 – SingleKarma

Input:

entity_sti – the symbolic time intervals of the single entity; that is, e.I

T – a [full] set of TIRPs that were discovered in the training phase;

Epsilon – the size of epsilon in the temporal relations;

Output: Single_T – an enumerated tree of TIRPs

1. *Single_T* ← Ø
2. Foreach $I^i, I^j \in$ entity_stis $\wedge$ $i<j$;
3. $r \leftarrow$ the temporal relation among I^i, I^j given *epsilon*
4. if $< I_{.sym}{}^i, r, I_{.sym}{}^j > \in T^2$;
5. Index(*Single_*T^2, $< I_{.\mathrm{sym}}{}^i, r, I_{.sym}{}^j >$) *//indexed as in Algorithm 2*
6. End Foreach
7. return *Single_T*; *//return a tree of up to two levels*
8. End

Algorithm 6 is similar to algorithm 3 (Lego), but instead of generating all of the frequent TIRPs, it generates the TIRPs that appear in T, which were discovered in the training phase and are required for the classification, and searches for them in the second level (Single_T2) that was created in SingleKarma.

The extended TIRP *t* is an object containing the TIRP properties, such as: a vector of symbols, a vector representing the half-matrix of the temporal relations, and a list of supporting instances (which is basically a vector of pointers to the sequence of time intervals), as described in more detail elsewhere (Moskovitch and Shahar, In Press). The TIRP's object *t* data structure supports a highly efficient O(1) time query regarding any particular relationship between two symbols in Single_T[2], without starting the detection of the TIRP within the entity's set of symbolic interval from scratch, as a naïve TIRP detection method might do.

If supporting instances of a TIRP are found, SingleLego searches for its extended TIRPs, as long as they appear in T, recursively. Thus, in each extension, new TIRPs are created using an extended vector of symbols (i.e., adding the next, k^{th} symbol), a vector of temporal relations (i.e., the temporal relations between the candidate symbolic time interval and the previous (*k*-1) symbolic time intervals), and the list of supporting instances. This list's vertical support can only stay the same, or decrease, as the TIRP is being extended into increasingly specific patterns, although its horizontal support can increase, depending on the particular set of symbols prevalent for each entity.

Algorithm 6 – SingleLego

Input:

Single_T – the enumerated tree of the entity (sent by reference)

T – the enumeration tree of the training set,

t – the extended TIRP;
level – the level of the enumeration in the tree*epsilon* – the size of the epsilon

Output: void

1. Foreach $t^{level+1} \in T$
   (Note that like *t*, $t^{level+1}$ is an object that stores, for any two symbols, all of the symbolic interval instances satisfying all of the relations between them)
2. $t^{level+1}.insts \leftarrow$Search supporting instances of $t^{level+1}$ in *Single_T*
3. if |inst| > 0 *//found at least one instance of this level in T, within Single_T*
4. SingleLego(*Single_T, T*, $t^{level+1}$, level+1, *epsilon*) *//extend by another level*
5. End Foreach
6. End

The single mining process results with an enumeration tree Single_T, which is a sub-tree of T, containing all of the TIRPs that were discovered in the given entity *e*. Then, as we show in Section 4, a vector of features (TIRPs), having one of the TIRP representations, is created. The vector of features is then used for classification. Note that each TIRP of size $k \geq 2$ is considered a feature; i.e, not just the longest TIRPs, or the leaves of the TIRP tree.
The SingleKarmaLego algorithm is actually significantly superior, computationally, to a naïve method that attempts to find each feature (i.e., TIRP) within a vector of symbolic time interval instances of the given entity (sorted lexicographically). The advantage can be demonstrated even when searching only for 2-sized TIRPs, and

looms much larger as the entity is queried for longer and longer TIRPs within the SingleLego recursive procedure.

The first and main advantage of the SingleKarmaLego algorithm stems from the one-time, preprocessing indexing step performed in SingleKarma, in which the Single_T^2 tree, representing all of the relations between two symbolic time intervals that appear in the entity, is created, represented as a matrix indexed by the two symbols. The result is a highly efficient structure for the purpose of query and retrieval of 2-sized TIRP instances, which supports, during the TIRP extension process, a very fast (O(1)) way of determining whether a given relation indeed exists between symbolic time interval instances of the two symbols.

The second advantage of SingleKarmaLego stems from the fact that, since a tree of TIRPs is created for the single entity's interval instances, adding the *k*th symbol to a (*k*-1) sized TIRP exploits the data structure already created for the (*k*-1) sized TIRP, to quickly check whether the *k*th symbol can indeed to be added to the list of feature TIRPs detected in the entity's record.

Let us denote by *m* the number of TIRPs discovered during the training phase, which need to be detected within the current set of *n* symbolic time intervals of the entity; and by *k* the number of symbols in a TIRP to be detected within the entity.

To compute complexity, we will count the number of operations that are needed to perform each TIRP-query (i.e., each search for a given TIRP within the entity's record), an operation being one of a computation of the temporal relation between two given time-stamped intervals, a single lookup into a table, a hash function call, or moving one step along a vector of intervals and looking up the contents indexed by that position.

In the simplest case, when k=2, The SingleKarma algorithm requires a constant preprocessing one-time set-up cost of $n^2/2$, or $O(n^2)$ operations to index *all* pairwise relations in the entity *n* symbolic time intervals; query and retrieval for a given symbols pair and temporal relation (i.e., determining if there is such an instance, and if so, retrieving at least one example) requires only O(1) operations, by looking up the index of Single_T^2 of each of the *m* queries, is then performed at m*O(1) time.

Thus, when k=2, the cost for SingleKarmaLego of *m* TIRP-queries is $O(n^2)+m*O(1)$.

Versus that, even for k=2, a naïve algorithm requires $O(n)$ operations to find or prove the nonexistence of a single 2-sized TIRP (two related symbolic time intervals) within a vector of time intervals, even though the intervals are sorted lexicographically.

Thus, a naïve algorithm requires even for k=2 symbols per TIRP, $m*O(n)$ operations for *m* TIRP-queries.

Since typically $m>n$ and usually grows much faster as the number of symbolic concepts grows (*m* grows exponentially as the number of symbols *s* grows, while for a given segmentation of the data into time intervals, the number of symbolic intervals *n* grows linearly with *s*), the preprocessing step leads to a significant advantage that grows as *m* grows.

When k>2, the advantage of SingleKarmaLego looms much larger, both due to the key pre-processing step, which creates the Single_T^2 and reduces the cost of each pair-query, and due to the efficiency of the SingleLego procedure itself. That is because, in general, for a k-sized TIRP, for a naïve algorithm, we need to perform for each of the m TIRP-queries, (k(k-1)/2) operations (one for each pairwise temporal relation) to disambiguate all of the pairwise relations among all components of the given TIRP.

Overall, for any k, m*(k(k-1)/2)*O(n) operations for m TIRP-queries, for a naïve algorithm.

Note that, apart from its inefficiency in checking each pairwise relation between two symbols (i.e, asking a size-2 TIRP-query), a naïve algorithm also does not exploit the fact that a TIRP-query was previously asked about a (k-1) sized TIRP included within the k-sized TIRP.

But in SingleLego, both of these advantages prove their benefit.

First, verifying that a pair of symbolic interval instances exists for each pairwise relation between two symbols that is checked while extending a given TIRP, requires only O(1), possibly incremented by a hash-function call to search among the several instances of the given symbol pair, all indexed by the same entry in the matrix of symbolic pairs created in the SingleKarma phase. This is because such an operation looks up, as explained, the array of pairwise relations, Single_T^2.

Second, for each k-sized TIRP, only (k-1) relations need to be checked, comparing the new symbol at level k to the previous (k-1) symbols (as illustrated in the last column of the half-matrix in figure 5). Note that, as a result of the depth-first recursive process, when the k^{th} symbol of a k-sized TIRP (feature) is added and its pairwise relations with the rest of the (k-1) symbols are being queried for existence in the entity's record, the specific corresponding (k-1) sized TIRP (including the specific symbolic interval instances from which it is composed) is being held in memory, so the only checks needed to be made consist of comparing the new symbol to the previous (k-1) symbols, in the context of these particular (k-1) symbolic-interval instances. As explained, each such check exploits the Single_T^2 data structure and requires only O(1) operations.

Overall, for any k, O(n^2) + m*(k-1)* O(1) operations are needed for m TIRP-queries, for SingleKarmaLego.

To sum up, naïve search requires approximately O(n)*m*k^2 operations, while SingleKarmaLego requires approximately only O(n^2) + m*k operations, a highly significant computational advantage that looms larger as the number of TIRP-queries, m, increases (exponentially, relative to the number of symbolic concepts), and as the number of symbols per TIRP, k, increases (note that k is really limited only by the number of intervals within the entity of the training set that has the maximal number of intervals).

To complete the TIRP-detection phase within the single entity, once all of the TIRPs of size k≥2 are detected within the single entity, their horizontal support and mean

duration are calculated as part of their feature representation, for the classification process.

4 Evaluation

The objective of the evaluation of the KLS framework, whose goal is to use the discovered TIRPs for the purpose of classifying multivariate time series, was to assess the feasibility and general effectiveness of using TIRPs for classification, and to assess the optimal settings of the method's various parameters (settings). Since the KLS framework includes quite a few potentially meaningful parameters, the number of required experiments was large.

The *research questions* posed were related to the varying parameters within the KLS framework (see additional details in the descriptions of the experimental design and of the results):

A. Which *state abstraction method* is the best for classification purposes?
   The goal of this question was to evaluate the EWD and SAX computational state abstraction methods suggested in this paper, in addition to the knowledge-based temporal-abstraction.

B. What is the *best number of bins* for classification purposes?
   The number of bins, which is the number of states in the state abstraction, determines the level of granularity of the abstraction method. Two options were evaluated: three and four states for the various abstraction methods.

C. Which set of temporal relations is the best for classification purposes?
   Two sets of temporal relations were evaluated in the KarmaLegoS framework, as defined in definition 1 in section 3.2: the full set of Allen's seven unique temporal relations, and a more abstract set of three temporal relations (BEFORE, OVERLAPS, CONTAINS).

D. Which size of epsilon is the best for classification purposes?
   The size of the epsilon parameter (see Definition 1 in Section 3.2) determines the flexibility of the temporal relations. Five general epsilon values were investigated: epsilon = 0, which is the default crisp version of Allen's temporal relations, versus a larger size epsilon: 1, 3, 5, and 10 temporal units.

E. Which TIRP representation is the best for classification purposes?
   The representation of a TIRP value for an entity in the training and test phases included three options: binary (i.e., whether the TIRP exists at least once in the record) (B), horizontal support (HS) and mean duration (MeanD).

F. Which feature selection measure is the best for classification purposes?
   As part of the training phase and the tabular construction, a feature selection filter was applied, including GainRatio and GAIN with several sets of features for comparison. The first objective was to determine whether the GAIN measure (Patel, 2008) is better than a common feature selection measure, such as GainRatio, in addition to the use of no feature selection.

G. Which classifier-induction method is the best for generating a TIRP-based classifier?
Several classifier-induction methods were used in the experiments, including the standard methods: Random Forest and Naïve Bayes, and the IE_Classifiers proposed by Patel et al (2008). For the comparison, only the Binary TIRP representation was used, in order to compare these classifiers to the standard classification algorithms. Note that in this evaluation, our objective was not to determine the optimal classifier-induction method; we simply wanted to compare the IE classifiers to a small number of common, generic induction algorithms, since we thought that standard existing classifiers might well suffice. Once we determined that the Random Forest method was superior to the other methods we used, in most of the first experiment, using a binary representation, it was used in the other two experiments, to evaluate the effect of the number of temporal relations, the various TIRP representations, and the value of the Epsilon parameter.

4.1 Datasets

4.1.1 The Diabetes Dataset

The diabetes dataset, provided as part of collaboration with Clalit Health Services (Israel's largest HMO) contains data on 2004 patients who have type II diabetes. The data were collected each month from 2002 to 2007. The dataset contains six concepts (laboratory values or interventions) recorded over time for each patient: hemoglobin-A1c (HbA1C) values (indicating the patient's mean blood-glucose values over the past several months), blood glucose levels, cholesterol values, and several medications that the patients purchased: oral hypoglycemic agents (diabetic medications), cholesterol reducing statins, and beta blockers. The total amount of the diabetic medications was represented in the dataset in terms of an overall defined daily dose (DDD). Knowledge-based state-abstraction definitions for abstraction of the raw-data laboratory-test measurements into more meaningful concepts were provided by expert physicians from Ben Gurion University's Soroka Medical Center. The diabetes dataset was used for the runtime evaluation and for the clustering experiments.

For the classification experiments, *determining the gender of the patients was used as the classification target*, since no clinical end points were provided in this particular dataset. Since there were 992 males and 1012 females, the dataset was quite balanced. Table 1 contains the cutoff definitions used for each state in the case of the raw measurements included in the Diabetes dataset. The rest of the raw data consisted of medications, for each of which a DDD abstraction was defined.

Table 1 Laboratory-measurement data types and their cut-off (discretization) values for the knowledge-based state abstractions defined for each, in the case of the diabetes data set.

Blood Glucose (mg/dL)	
State	**Glucose**
1	< 100
2	100-125
3	126-200
4	>200

Hemoglobin A1C (%)	
State	**HbA1c**
1	< 7
2	7-9
3	9-10.5
4	>10.5

LDL Cholesterol (mg/dL)	
State	**LDL**
1	< 100
2	100-130
3	130-160
4	> 160

HDL Cholesterol (mg/dL)		
State	**HDL-Male**	**HDL-Female**
1	< 35	< 30
2	35 – 45	30 – 40
3	> 45	> 40

4.1.2 The Intensive Care Unit (ICU) Dataset

The ICU dataset contains multivariate time series of patients who underwent cardiac surgery at the Academic Medical Center in Amsterdam, the Netherlands, between April 2002 and May 2004. Two types of data were measured: static data including details about the patient, such as age, gender, type surgery the patient underwent, whether the patient was mechanically ventilated more than 24 hours during her postoperative ICU stay, and high frequency time series, measured every minute over the first 12 hours of the ICU hospitalization.

The data include: mean arterial blood pressure (ABPm), central venous pressure (CVP), heart rate (HR), body temperature (TMP), and two ventilator variables, namely fraction inspired oxygen (FiO2) and level of positive end-expiratory pressure (PEEP), and low frequency time-stamped data, including base excess (BE), cardiac index (CI), creatinine kinase MB (CKMB) and glucose. For the knowledge-based state abstraction, we used the definitions of Verduijn et al (2007). In addition, the data were abstracted with computationally unsupervised and supervised abstraction methods. (See the experimental results section).

The dataset contains 664 patients; 196 patients were mechanically ventilated for more than 24 hours. 19 patients had very few values and were removed. Thus, the experimental data set included 645 patients of whom 183, or 28%, were mechanically ventilated for more than 24 hours. *The main classification goal was determining if the patient needed ventilation after 24 hours, given the data of the first 12 hours.*

4.1.3 The Hepatitis Dataset

The hepatitis dataset contains the results of laboratory tests performed on patients who hepatitis B or C, who were admitted to Chiba University Hospital in Japan. Hepatitis A, B, and C are viral infections that affect the liver of the patient. Hepatitis B and C chronically inflame the hepatocyte, whereas hepatitis A acutely inflames it. Hepatitis B and C are especially important, because they involve a potential risk of developing liver cirrhosis and/or carcinoma of the liver.

The dataset contained time-series data regarding laboratory tests, which were collected at Chiba University hospital in Japan. The subjects included 771 patients of hepatitis B and C who had tests performed between 1982 and 2001. The data included administrative information, such as the patient's demographic data (age and date of birth), pathological classification of the disease, date of biopsy, result of the biopsy, and duration of interferon therapy. Additionally it included the temporal records of blood tests and urinalysis. These tests can be split into two sub-categories, in-hospital and out-hospital test data. In-hospital test data contained the results of 230 types of tests that were performed using the hospital's equipment. Out-hospital test data contain the results of 753 types of tests, including comments of staff members, performed using special equipment at the other facilities. Consequently, the temporal data contained the results of 983 types of tests. We selected eleven variables which were found most frequent (occurring in most of the patients), including: Glutamic-Oxaloacetic Transminase (GOT), Glutamic-Pyruvic Transminase (GPT), Lactate DeHydrogenase (LDH), TP, ALkaline Phosphatase (ALP), Albumin (ALB), Total BILirubin (T-BIL), Direct BILirubin (D-BIL), Indirect BILirubin (I-BIL) and Uric Acid (UA). Many patients had a limited number of tests, so we focused only on the variables occurring frequently, which included 204 patients who had Hepatitis B and 294 patients who had Hepatitis C. No knowledge-based abstractions were available for that domain, so only automated abstraction methods were used. *The objective was to classify patients as having either Hepatitis B or C.*

4.2 Experimental Design

The evaluation of the KLS framework focused on the various parameters of the framework as presented earlier, by performing three key experiments, all using the discovered TIRPs as classification features, which together answer all of our research questions.

Experiment 1: Varying the number of state-abstraction bins, type of feature selection method, and type of classification method.

We evaluated three state-abstraction methods: knowledge-based (KB), EWD, and SAX; the number of bins used for automated discretization was either 3 or 4. We also wanted to compare the GAIN measure suggested by Patel (2008) to the standard GainRatio measure as a feature selection method. To be able to compare the two measures we set GAIN to 0.02, as was recommended by Patel, and used GainRatio with the same number of features. Additionally, we used both measures with the same number of 100 top preferred features for comparison. We used the option of no feature selection, which we called NoFS, as a baseline.

Finally, we also compared the performance of the standard classification algorithms RandomForest and Naïve Bayes, to that of the IE Classifiers.

Experiment 2: Varying the number of temporal relations and the TIRP representation method.
The main research question here focused on the use of temporal relations: the use of Allen's seven temporal relations versus the use of only the three abstract relations, as presented in definition 1. Additionally, we compared the two novel TIRP representations, HS and MeanD, to the default Binary (existence) representation.

Experiment 3: Varying the epsilon value.
The research question here focused on the influence of the size of the Epsilon on the classification performance.

To answer the various research questions we ran a set of evaluation runs with 10-fold cross validation. Note that, as explained in Section 3.4.1, during the cross validation, in each iteration, the mining phase was repeated on the relevant data set to extract the relevant features, and the rest of the steps were performed separately. We used a 60%, 40% and 60% minimal vertical support threshold for the Hepatitis, Diabetes and ICU12h datasets respectively. Determining the minimal vertical support is an important issue, since having a low threshold will discover a larger number of TIRPs, but might require a much longer runtime. Thus, we used thresholds that will discover TIRPs for all the experimental settings and that will lead to ending the discovery process within a reasonable time.

4.3 Results

4.3.1 Experiment 1

The first experiment focused on determining the best number of bins and comparison of the GainRatio and the GAIN feature selection measures and the RandomForest and Naïve Bayes standard classifiers to the IE Classifiers. To compare the feature selection measures we used two options: we set GAIN to 0.02, which we termed GAIN002, and used GainRatio with the same number of top selected features (as at GAIN002), which we termed GR002; and we used both measures with the top 100 selected features, which we termed GAIN100 and GR100. To generate a baseline, we ran the evaluation also without feature selection, which we termed NoFS.

To decrease the settings' complexity and isolate the explored settings, we set the other settings to their default values: Temporal relations number = 7; Epsilon value = 0; TIRP representation = Binary; Feature selection = Gain002, GR002, Gain100, GR100 and NoFS; Classifiers = Random Forest, Naïve Bayes, IE_Confidence, IE_Majority. Since we wanted to evaluate also the various state abstraction methods and number of bins we set them to: Abstraction methods = KB, EWD, and SAX; Number of bins for automated discretization = 3 and 4. We ran the experiment on each of the datasets using 10-fold cross validation.

Figures 10 to 12 show the results of experiment 1 on each of the datasets, **thus, each result is the mean of 200 evaluation runs**; the mean results of all the datasets are shown in figure 13, **and each result plotted is based on 600 evaluation runs**. In the

case of the Diabetes dataset, using the KB abstraction method led to much better performance than using the SAX and EWD methods, while SAX with four states led to the best performance amongst the automated methods. For the ICU12h dataset, use of the EWD method produced better performance than the use of SAX, for either three or four bins, and use of the KB definitions resulted in a performance similar to using EWD with three bins. For the Hepatitis classification task, SAX performed better than the EWD method, and using four states resulted in better performance than using three states.

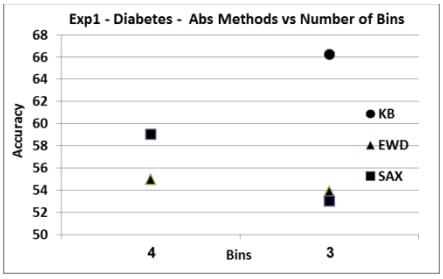

Fig 10. The KB abstraction performs better than the EWD and SAX, and using 4 states was better than 3, in the case of the Diabetes domain task.

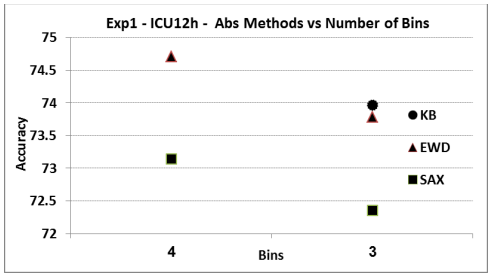

Fig 11. EWD performed better than SAX, and using 4 states was better than using 3, in the case of the ICU domain task.

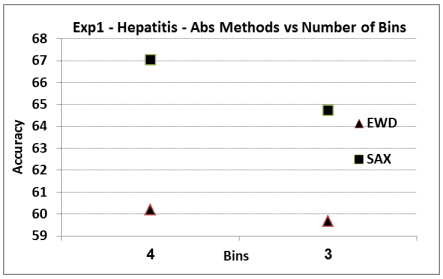

Fig 12. SAX performed better than the EWD method, and using 4 states resulted in better performance than using 3 states, in the case of the Hepatitis domain task.

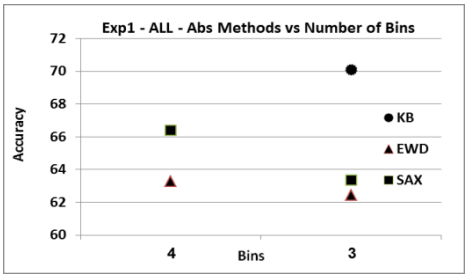

Fig 13. On average, across all relevant domains and experimental settings, the KB abstraction method led to a superior performance, while SAX performed better than EWD when using either 3 or 4 bins.

The mean results, averaged across all of the experimental variations (Figure 13) show that using SAX led to better performance than using EWD, using either three or four bins, while using a KB abstraction (based on two datasets) led to better performance than using the computational methods. Note that each result is the mean of 200 evaluation runs, due to the various runs of the four classifiers and five features selection measures, as detailed in the next results.

Figures 14 to 17 show the results of the classifiers and the feature selection measures of Experiment 1 on each of the datasets, and the mean results of all the datasets. **Each result in figures 14 to 16 is the mean of 40 evaluation runs, and each result in figure 17 is the mean of 120 evaluation runs**. In the Diabetes dataset, the Random Forest and Naïve Bayes induction methods performed similarly, both outperforming the IE Classifiers. The feature selection methods performed similarly, including the NoFS. In the ICU12h task, the Random Forest induction method outperformed the other classifiers. The feature selection methods performed similarly, while the Naïve Bayes method performed worst when using the Gain with the top 100 features. In the Hepatitis domain, the Random Forest method outperformed the other classifiers, and the Naïve Bayes classifier was slightly better than the IE Classifiers. Again, none of the feature selection methods seemed very useful, thus making our original research question a moot one, although using GR100 proved to be the best for the IE Classifiers and the Naïve Bayes methods.

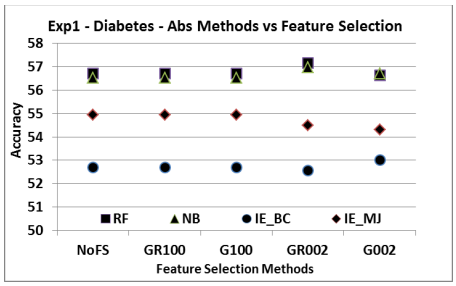

Fig 14. The Random Forest (RF) and Naïve Bayes (NB) methods performed similarly on the diabetes data set, outperforming the IE classifiers. The feature selection method (if any) did not have much effect.

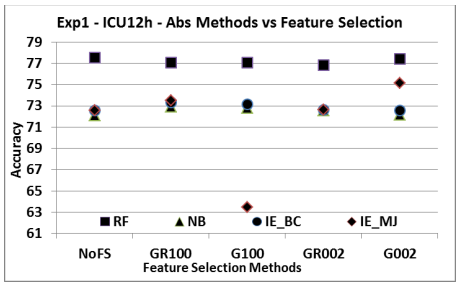

Fig 15. The Random Forest (RF) method outperformed the other classifiers in the ICU domain. The feature selection methods mostly did not seem to be useful.

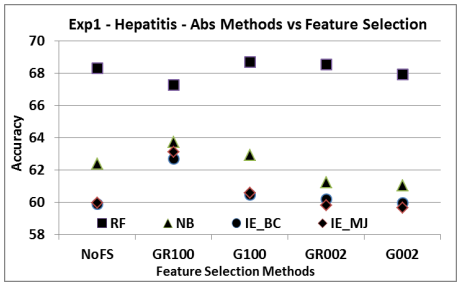

Fig 16. The Random Forest (RF) method outperformed the other classifiers in the Hepatitis domain. The GR100 feature selection method performed best for the classifiers, except for RF, in which it was slightly less effective.

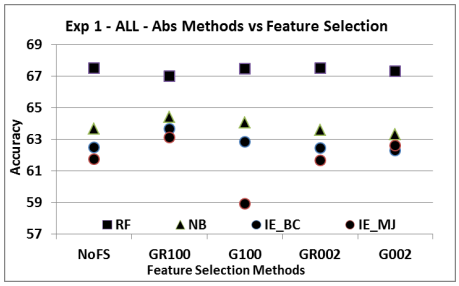

Fig 17. Mean results across all data sets and settings. The Random Forest (RF) method outperformed the other classifiers, and the feature selection methods didn't seem to contribute to the results.

The mean results across all the datasets show that the Random Forest induction method outperformed the rest of the classifiers; and that the feature selection methods were not useful. The low effectiveness of the feature selection can be explained by the relatively low number of TIRPs (features) discovered originally.

4.2.1 Experiment 2

In experiment 2 we wanted to evaluate the performance of the temporal relations sets: Allen's original set of seven temporal relations versus the set of three abstract temporal relations that are proposed in definition 1, and to evaluate the additional two TIRPs (features) representations: the HS (definition 7) and the MeanD (definition 9) in comparison to the default Binary representation.

The settings of this experiment were: Abstraction methods: KB, EWD and SAX; Number of Bins: 3 and 4 bins; Temporal relations: Allen's seven relations (7) and our three abstract temporal relations (3); the epsilon value was set to 0; TIRP representation methods: HS and MeanD, in addition to the Binary representation. Given the results of experiment 1, we used here only the RandomForest classifier, which proved superior to the other induction methods, and used no feature selection (NoFS), since no clear value was demonstrated for any feature selection method.

Figures 18 to 21 show the effect of varying the number of temporal relations versus varying the TIRP representations. **Each result in figures 18 to 20 is the mean of 70 evaluation runs and in figure 21 is the mean of 210 evaluation runs**. The results in the case of the Diabetes classification task show that using three abstract relations outperformed using all seven relations for any type of TIRP representation, which in fact didn't demonstrate any meaningful influence on the performance. In the ICU12h classification task, using three temporal relations led to a better performance and the MeanD representation performed better than the Binary, which was better than the HS. In the Hepatitis classification task, using the set of three temporal relations was better than using the set of seven relations. The MeanD and HS representations performed better than the Binary TIRP representation; the HS representation performed better than MeanD, especially when using seven relations.

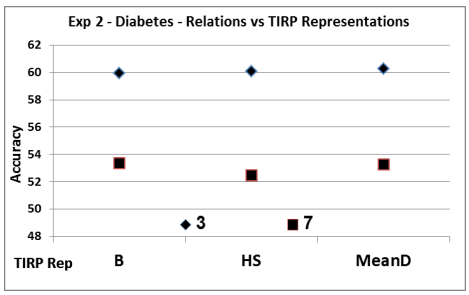

Fig 18. The use of 3 temporal relations outperformed the use of 7 temporal relations, in the diabetes domain; the MeanD feature representation method was marginally better than the HS and Binary representations.

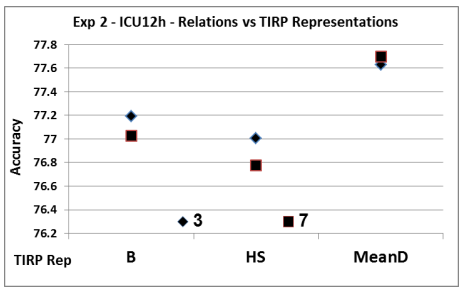

Fig 19. The use of 3 temporal relations was slightly better than the use of 7 temporal relations, in the case of the ICU domain, while the MeanD representation was better than the Binary one, which was better than using the HS.

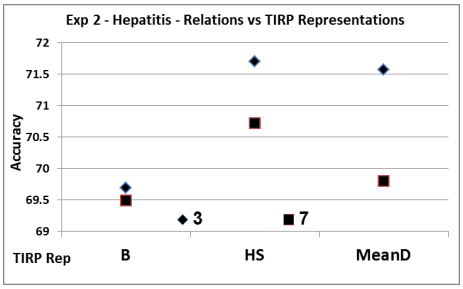

Fig 20. The use of 3 temporal relations was better than the use of the 7 temporal relations, in the case of the Hepatitis domain, while the HS representation method was best.

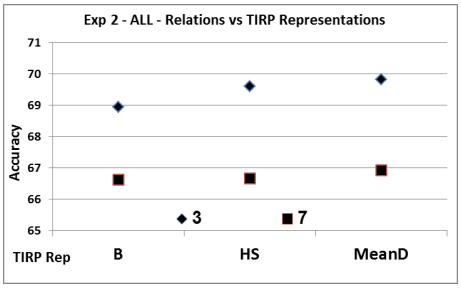

Fig 21. Mean results across all data sets and settings. The use of 3 temporal relations was slightly superior to the use of 7 temporal relations, while the MeanD TIRP representation performed somewhat better than HS, which was somewhat better than the Binary representation method.

The mean results across all of the datasets and settings show that using three temporal relations was superior to the use of seven temporal relations; and that the MeanD TIRP representation method performed somewhat better than the HS representation, which was somewhat better than the Binary representation.

Thus, we conclude that the use of three abstract temporal relations is better for the purpose of classification than the use of Allen's seven relations. The MeanD and HS TIRP representations are better than the default Binary representation. These two measures are unique to the KarmaLego framework, which, unlike common mining methods, discovers all of the instances of a pattern within the same entity, and doesn't stop at the first instance discovered (thus enabling it to calculate the horizontal support and the mean duration within each entity), which is enabled by its efficient candidate generation and computation.

4.2.1 Experiment 3

Experiment 3 evaluates the influence of the epsilon value on the classification accuracy. To isolate that influence, we used the following settings: as abstraction methods we used EWD and SAX, both using 3 bins. We used 3 bins for the analysis, since when using 4 bins we didn't discover a sufficient number of TIRPs for some of the epsilon values. The TIRP representation method was set to MeanD; we used NoFS – that is, no feature selection; and the induction method was Random Forest.

Figures 22-25 show the results of Experiment 3, based on the mean of 10 fold cross validations (figure 25 results are the mean of 30 evaluation runs). We used for the epsilon values the following values: 0, 5, 10, 20 and 50 for the ICU12h and the Hepatitis datasets. In the Diabetes domain, we used 0 as the default, 1, 2, 3, 5, and 10. We didn't use 20 and 50, since the dataset contained a maximum of 60 time points for each patient, but instead we used 1 and 3.

The performance of the abstraction (discretization) methods in this experiment was the same as before; thus, SAX outperformed EWD most of the time, except for a mixed result in the case of the ICU12h dataset. In the Diabetes dataset increasing the

Epsilon values improved slightly the performance for the SAX method, but the opposite happened for the EWD. In the ICU12h dataset increasing the Epsilon value decreased slightly the performance for SAX, and didn't change much for the EWD, although for Epsilon = 5 it improved. In the Hepatitis dataset, increasing the size of the Epsilon increased the accuracy when using the SAX method, but did not change it for the EWD method.

The mean results of the all the datasets indicate that increasing the value of the Epsilon decreased accuracy when using the EWD discretization method, while slightly increasing when using SAX. Thus, we conclude that increasing the Epsilon value is, in general, not recommended for purposes of an accurate classification.

In addition to the Accuracy measure, we measured the True Positive Rate (TPR) and True Negative Rate (TNR) for the minority class in each domain. We observed for SAX in the Diabetes dataset a TPR in the range of 57.8% to 59.5%, and a TNR in the range of 56.4% to 59.2%; in the ICU12h dataset, a TPR in the range of 89.5% to 93.5%, and a TNR in the range of 35.9% to 45.9%; and in the Hepatitis dataset, a TPR in the range of 85.5% to 89.7%, and a TNR in the range of 60.5% to 70 %. Across all of datasets, the mean TPR was in the range of 77.6% to 80.9% and the mean TNR was in the range of 50.9% to 58.4%.

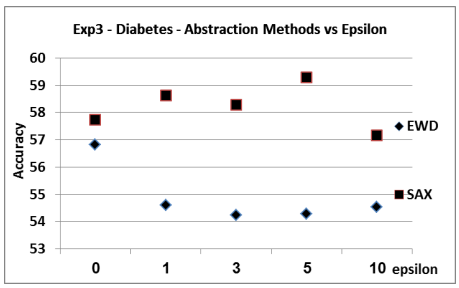

Fig 22. In the diabetes domain, the use of a larger epsilon usually reduced accuracy when using the EWD discretization method, while usually improving it slightly when using the SAX method.

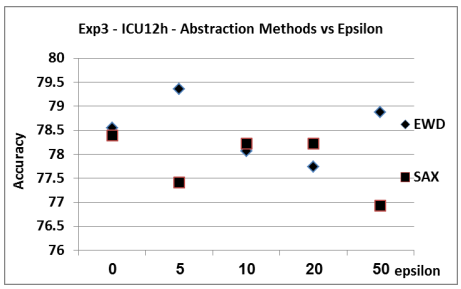

Fig 23. In the ICU domain, the use of a larger epsilon had an inconsistent effect on accuracy; it usually led to lower performance, in the case of the SAX method.

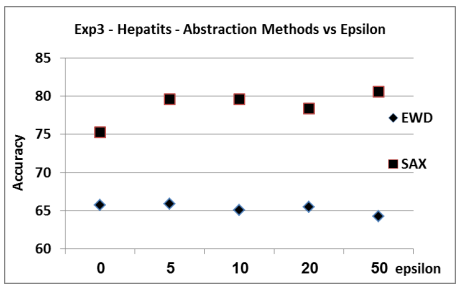

Fig 24. In the Hepatitis domain, the use of a larger epsilon usually increased accuracy slightly when using the EWD method, while having little effect when using the SAX method.

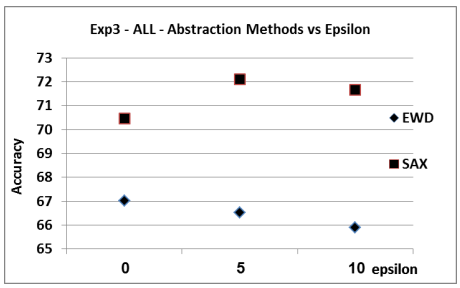

Fig 25. The mean overall results across all data sets and settings show that using a larger epsilon tends to lead to a slightly better accuracy for SAX, but to a reduced accuracy when using EWD.

5 Discussion and Conclusions

In this study, we have introduced a comprehensive architecture and process, *KarmaLegoSification* (KLS), for classification of multivariate time series, which is based on a temporal-abstraction process that can exploit either domain knowledge, when provided, or, when needed, perform on-the-fly discretization, and which introduces several fundamental concepts that define the output of the time intervals mining process. The KLS framework uses *KarmaLegoSingle* - an algorithm for fast TIRPs mining in a single entity that is represented by symbolic time intervals, based on the principles underlying the KarmaLego algorithm for fast temporal-pattern mining (Moskovitch and Shahar, In Press). KarmaLegoSingle is used for the detection of TIRPs in the classification phase. Since in a single entity the notion of minimal vertical support is meaningless, the detection task is performed by mining only the TIRPs that were discovered as frequent in the training set and used in the classification model.

Mining time intervals patterns is computationally highly demanding, but this is true in general for multivariate time series mining. Transforming the time series into time intervals series reduces significantly the amount of data to analyze. The KarmaLego algorithm for fast temporal-interval pattern mining is briefly introduced in this paper and appears in more detail elsewhere (Moskovitch and Shahar, In Press). The algorithm performs temporal-interval pattern mining efficiently, mostly due to its data structure and to its efficient candidate generation, which exploits the transitivity property of temporal relations.

In this paper, we have introduced as one of its contributions the SingleKarmaLego algorithm, which enables us to detect a set of TIRPs (used here as classification features) that appear within a single entity's set of symbolic time intervals. We analyzed the complexity of SingleKarmaLego, demonstrating its superior efficiency when compared to a naïve TIRP detection method. Due to the SingleKarma phase, in which the symbolic time intervals are indexed, we can later access the pairs of intervals for a given relation in O(1) instead of O(n); and due to the nature of the Lego method, which stores all patterns and their sub-patterns in the same data structure, the algorithm requires only k-1 operations for each k-sized TIRP detection, since the (k-1) sized TIRP contained within it was already indexed; while a naïve detection method requires $k(k\text{-}1)/2$ operations to examine all of the possible temporal relations for each unambiguous TIRP being searched in the entity's record. Although the SingleKarmaLego method does require a one-time preprocessing cost of O(n^2), due to the SingleKarma phase that indexes all of the pairwise relations in the entity's record, it is still significantly superior to a naïve method that attempts to detect each TIRP on its own by querying the single entitiy's record for it; its benefits increase as the overall number of different symbolic concepts in the training set increases, and as more symbols are used within the TIRPs used as features.

Our evaluation was not designed to examine or enhance the accuracy measures per se, but rather, to investigate the feasibility of the overall classification approach, using only TIRPs, and, in particular, to answer the specific research questions we raised.

As part of our rigorous evaluation, which was performed in three data sets originating from three highly different medical domains (unlike most previous studies, which had typically used just a single dataset), we have quantitatively evaluated the difference between the use of Allen's original basic seven temporal relations, versus the use of an abstract version of three temporal relations that we proposed here. Note that the proven effectiveness of using only three (abstract) relations greatly reduces the computational complexity of the TIRP enumeration phase in the core KarmaLego algorithm, which is highly (exponentially) sensitive to the number of temporal relations. In future research, one could examine even the effect of using only two abstract relations, such as *Before* and *Overlaps* (i.e., the rest of Allen's relations), as suggested by some authors (Batal et al, 2012) (better terms might be *Before* and *Coincide*), or focus on the relation *Precedes* (Before, Meets, or Overlaps) versus the rest of Allen's relations, as suggested by other authors (Sacchi et al, 2007), etc.

With respect to the abstraction method, usually, the SAX discretization method, either using 3 or 4 bins and considering all of the test domains, proved superior to the EWD method, which is a popular default, although, when available, the knowledge-based abstraction method was, on average, quite superior to both, in particular in the diabetes domain. This is a somewhat surprising observation, since it is certainly not obvious that domain experts (clinicians, in this case) necessarily create abstractions that are useful not only for their daily use, such as for continuous monitoring or therapeutic (diagnostic) purposes, but also for (possibly future, or external) classification purposes (Verduijn et al, 2007). However, these results suggest that use of domain abstraction knowledge must always be considered at least as an option, when available.

For these particular data sets, the Random Forest induction method proved superior to the other methods we experimented with, including other methods reported in a similar context of temporal data mining, and was thus the default method for the second and third experiments.

As another part of the evaluation, we have introduced and assessed two novel representation methods for temporal features (i.e., TIRPs), which exploit the capability of the KarmaLego framework to compute the number and the mean duration of multiple instances of discovered TIRPs within each time-oriented record, in addition to a Binary, or Boolean (purely existential) representation, for the purpose of classification. These representation methods were shown to have an added valued, when compared to the default Binary representation method. This enhanced representation is achieved without requiring any additional computational cost, since, as pointed out in the comments to the Lego algorithm, to preserve the completeness of the TIRP discovery process, i.e., in order to discover *all* of the different TIRPs within an entity, whether we need one or more instances of each TIRP, *all* of the instances of each TIRP have to be discovered within each entity, in any case.

We also conclude, given yet another aspect of the evaluation, that increasing the Epsilon value beyond 0 (the default value) is in general *not* recommended for purposes of classification, in particular when reducing the number of temporal relations to only three (thus already increasing flexibility). This conclusion might alleviate some potential difficulties, since determining the right Epsilon value might

be quite problematic and might in general require several trials for each new dataset. One possibility for explaining our results is that using a larger value for the epsilon parameter decreased the classification performance, since it decreased the number of frequent TIRPs discovered, which in turn, decreased the number of features and thus, eventually, the classification accuracy. Another possibility is that using an Epsilon value, although increasing robustness, has also resulted in somewhat more opaque (ambiguous) relations, thus perhaps losing some information that might be useful for classification purposes, and thus, overall, leading to little or no enhancement in accuracy.

One of the settings of the experiments which, to highlight the other aspects, we did not try to vary, is the minimal vertical support. We would like in future work to experiment with lower levels of minimal vertical support and evaluate their contribution. In addition, we did not use any single symbolic intervals as features (i.e., only TIRPs including two or more symbols). We will examine in the future the effect of including such single symbols as well. In some cases, a single interval of, say, "high blood pressure" might be sufficient as an important feature.

Overall, we conclude that TIRP-based classification, as performed within the KLS framework, is quite feasible, leads to a reasonable performance, and is potentially highly useful for classification and prediction purposes in time-oriented, multivariate-data domains. The enhanced performance achieved using a knowledge-based abstraction method in the case of the Diabetes dataset shows that better classification might be achievable through the use a better informed discretization method, although not necessarily based on domain knowledge, a direction that we plan to explore in our future work. Such a method is especially crucial in new domains, in which knowledge-based definitions do not necessarily exist.

Acknowledgements. The authors wish to thank Marion Verduijn for sharing the ICU12h dataset, and Prof. Avi Porath from the Soroka Academic Medical Center for his assistance regarding the diabetes data set. For insightful discussions time intervals mining and classification using TIRPs our thanks to Christos Faloutsos, Christian Freksa, Fabian Moerchen, Dhaval Patel and Iyad Batel. The authors also wish to acknowledge the useful comments of the anonymous reviewers, which have significantly improved this manuscript. This work was supported in part by grants from Deutsche Telekom Laboratories, HP labs Innovation Research Program.

References

J. F. Allen. Maintaining knowledge about temporal intervals, Communications of the ACM, 26(11): 832-843, 1983.

J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, Proceedings SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, July 2002.

R. Azulay, R. Moskovitch, D. Stopel, M. Verduijn, E. de Jonge, and Y. Shahar, Temporal Discretization of medical time series - A comparative study, IDAMAP 2007, Amsterdam, The Netherlands, 2007.

I. Batal, H. Valizadegan, G. Cooper and M. Hauskrecht, A Temporal Pattern Mining Approach for Classifying Electronic Health Record Data. ACM Transaction on Intelligent Systems and Technology (ACM TIST), Special Issue on Health Informatics, 2012

I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht. Mining Recent Temporal Patterns for Event Detection in Multivariate Time Series Data, Proceedings of *Knowledge Discovery and Data Mining (KDD)*, Beijing, China, 2012.

C. Freksa, Temporal reasoning based on semi-intervals, Artificial Intelligence, vol. 54, 1, 1992.

F. Höppner, Learning Temporal Rules from State Sequences, Proceedings of WLTSD, 2001.

F. Höppner, Time series abstraction methods - A Survey Workshop on Knowledge Discovery in Databases, Dortmund, 2002

B. Hu, Y. Chen, and E. Keogh, Time Series Classification under More Realistic Assumptions, Proceedings of SIAM Data Mining, 2013.

V. R. Jakkula, D. J. Cook, Detecting Anomalous Sensor Events in Smart Home Data for Enhancing the Living Experience, Artificial Intelligence and Smarter Living'11, 2011

P. S. Kam and A. W. C. Fu, Discovering temporal patterns for interval based events, In Proceedings DaWaK-00, 2000.

N. Lavrac, E. Keravnou and B. Zupan, Intelligent Data Analysis in Medicine, White Paper. Slovenia: Faculty of Computer Science and Information Sciences, University of Ljubljana, 2000.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu, A Symbolic Representation of Time Series with Implications for Streaming Algorithms, In 8th ACM SIGMOD DMKD workshop, 2003.

F. Mörchen, and A. Ultsch, Optimizing Time Series Discretization for Knowledge Discovery, In Proceeding of KDD05, 2005.

F. Mörchen, Algorithms for Time Series Knowledge Mining, Proceedings of KDD-06, 2006.

F. Moerchen, A better tool than Allen's relations for expressing temporal knowledge in interval data, Workshop on Temporal Data Mining, 2006.

F. Moerchen, and D. Fradkin, Robust mining of time intervals with semi-interval partial order patterns, Proceedings of SIAM Data Mining, 2010.

R. Moskovitch, D. Stopel, M. Verduijn, N. Peek, E. de Jonge, and Y. Shahar, Analysis of ICU Patients Using the Time Series Knowledge Mining Method, IDAMAP 2007, Amsterdam, The Netherlands, 2007.

R. Moskovitch, Y. Shahar, Medical Temporal-Knowledge Discovery via Temporal Abstraction, AMIA 2009, San Francisco, USA, 2009.

R. Moskovitch, N. Peek, Y. Shahar, Classification of ICU Patients via Temporal Abstraction and temporal patterns mining, IDAMAP, Verona, Italy, 2009.

R. Moskovitch, Y. Shahar, Fast Time Intervals Mining Using Transitivity of Temporal Relations, In Press for Knowledge and Information Systems.

P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos, Mining frequent arrangements of temporal intervals, Knowledge and Information Systems, 21 (2), 2009.

D. Patel, W. Hsu, M L Lee, Mining Relationships among Interval-based Events for Classification, Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008.

J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, Proc. 17th Int'l Conf. Data Eng. (ICDE '01), 2001.

L.R. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, Proceedings of the IEEE, 77, 1989.

C Ratanamahatana, E. J. Keogh, Three Myths about Dynamic Time Warping Data Mining, Proceedings of Siam Data Mining, 2005.

J. Roddick and M. Spiliopoulou, A survey of temporal knowledge discovery paradigms and methods. IEEE Transactions on Knowledge and Data Engineering, 4(14), 2002.

L. Sacchi, C. Larizza, C. Combi, and R. Bellazi, Data mining with temporal abstractions: learning rules from time series. Data Mining and Knowledge Discovery, (15), 2007.

G. Salton, A. Wong, and C.S. Yang, A vector space model for automatic indexing, Communications of the ACM, 18, 1975.

Y. Shahar, A framework for knowledge-based temporal abstraction, Artificial Intelligence, 90(1-2) 1997.

Y. Shahar, Dynamic temporal interpretation contexts for temporal abstraction, Annals of Mathematics and Artificial Intelligence 22 (1-2), 1998.

Y. Shahar, Knowledge-based temporal interpolation, Journal of Experimental and Theoretical Artificial Intelligence 11, 1999.

Y. Shahar, H. Chen, D. Stites, L. Basso, H. Kaizer, D. Wilson, and M.A. Musen, Semiautomated acquisition of clinical temporal-abstraction knowledge, Journal of the American Medical Informatics Association 6(6), 494-511, 1999.

M. Verduijn, L. Sacchi, N. Peek, R. Bellazi, E. de Jonge,B. de Mol. Temporal abstraction for feature extraction: A comparative case study in prediction from intensive care monitoring data, In Artificial Intelligence in Medicine, 41, 112, 2007.

R. Villafane, K. Hua, D Tran, and B. Maulik, Knowledge discovery from time series of interval events, Journal of Intelligent Information Systems, 15(1), 2000.

E. Winarko and J. Roddick. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data, Data and Knowledge Engineering, 1(63), 2007.

S. Wu, Y. Chen, Mining Non-ambiguous Temporal Patterns for Interval-Based Events, IEEE Transactions on Knowledge and Data Engineering, 19 (6), 2007.