Approximate Data Mining Techniques on Clinical Data

by

Matteo Mantovani

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science
Cycle XXXII/2016

at the

University of Verona

May 2020

Author . . . .
Department of Computer Science
Dec, 2019

Certified by . . . .
Prof. Carlo Combi
Full Professor
Thesis Tutor

Accepted by . . . .
Prof. Massimo Merro
Chairman
Council of the PhD School in Computer Science

Approximate Data Mining Techniques on Clinical Data

by

Matteo Mantovani

Submitted to the Department of Computer Science
on Dec, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science
Cycle XXXII/2016

Abstract

The past two decades have witnessed an explosion in the number of medical and healthcare datasets available to researchers and healthcare professionals. Data collection efforts are highly required, and this prompts the development of appropriate data mining techniques and tools that can automatically extract relevant information from data. Consequently, they provide insights into various clinical behaviors or processes captured by the data. Since these tools should support decision-making activities of medical experts, all the extracted information must be represented in a human-friendly way, that is, in a concise and easy-to-understand form. To this purpose, here we propose a new framework that collects different new mining techniques and tools proposed. These techniques mainly focus on two aspects: the temporal one and the predictive one. All of these techniques were then applied to clinical data and, in particular, ICU data from MIMIC III database. It showed the flexibility of the framework, which is able to retrieve different outcomes from the overall dataset.

The first two techniques rely on the concept of Approximate Temporal Functional Dependencies (*AT*-FDs). *AT*-FDs have been proposed, with their suitable treatment of temporal information, as a methodological tool for mining clinical data. An example of the knowledge derivable through dependencies may be *"within 15 days, patients with the same diagnosis and the same therapy usually receive the same daily amount of drug"*. However, current *AT*-FD models are not analyzing the temporal evolution of the data, such as *"For most patients with the same diagnosis, the same drug is prescribed* AFTER *the same symptom"*. To this extent, we propose a new kind of *AT*-FD called Approximate Pure Temporally Evolving Functional Dependencies (*APE*-FDs).

Another limitation of such kind of dependencies is that they cannot deal with quantitative data when some tolerance can be allowed for numerical values. In particular, this limitation arises in clinical data warehouses, where analysis and mining have to consider one or more measures related to quantitative data (such as lab test results and vital signs), concerning multiple dimensional (alphanumeric) attributes (such as patient, hospital, physician, diagnosis) and some time dimensions (such as the day since hospitalization and the calendar date). According to this scenario, we introduce a new kind of *AT*-FD, named Multi-Approximate Temporal Functional Dependency (*MAT*-FD), which considers dependencies between dimensions and quantitative measures from temporal clinical data. These new dependencies may provide new knowledge as *“within 15 days, patients with the same diagnosis and the same therapy receive a daily amount of drug within a fixed range”*.

The other techniques are based on pattern mining, which has also been proposed as a methodological tool for mining clinical data. However, many methods proposed so far focus on mining of temporal rules which describe relationships between data sequences or instantaneous events, without considering the presence of more complex temporal patterns into the dataset. These patterns, such as trends of a particular vital sign, are often very relevant for clinicians. Moreover, it is really interesting to discover if some sort of event, such as a drug administration, is capable of changing these trends and how. To this extent, we propose a new kind of temporal patterns, called Trend-Event Patterns (*TE*-Ps), that focuses on events and their influence on trends that can be retrieved from some measures, such as vital signs. With *TE*-Ps we can express concepts such as *“The administration of paracetamol on a patient with an* increasing *temperature leads to a* decreasing *trend in temperature after such administration occurs”*.

We also decided to analyze another interesting pattern mining technique that includes *prediction*. This technique discovers a compact set of patterns that aim to describe the condition (or class) of interest. Our framework relies on a classification model that considers and combines various predictive pattern candidates and selects only those that are important to improve the overall class prediction performance. We show that our classification approach achieves a significant reduction in the number of extracted patterns, compared to the state-of-the-art methods based on minimum predictive pattern mining approach, while preserving the overall classification accuracy of the model.

For each technique described above, we developed a tool to retrieve its kind of rule. All the results are obtained by pre-processing and mining clinical data and, as mentioned before, in particular ICU data from MIMIC III database.

Thesis Advisor: Prof. Carlo Combi
Title: Full Professor

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

This thesis deals with different approximate data mining techniques applied to clinical data. In particular, we focused on the problem of extracting insightful and unknown rules from clinical data and, for this purpose, we propose a new framework that groups different approximate data mining techniques. All these techniques produce a set of rules that should have to be interpreted by medical experts. Thus, it is important that all the extracted information is represented in a human-friendly way, that is, in a concise and easy to understand form. Moreover, we developed a tool for each technique described above in order to retrieve its kind of rule. Finally, these techniques have been applied by pre-processing and mining ICU data from MIMIC III database. In this introductory chapter, we will briefly introduce and motivate the focus of this dissertation, i.e., approximate data mining techniques on clinical data. Moreover, we will show why these techniques are relevant in real-world applications and how the temporal or predictive aspects of these techniques could increase their importance. Then, we will describe the original results presented in this thesis about the mining of clinical data. Finally, we will explain how this thesis is organized, and we will summarize the contribution of our previous publications.

1.1 Motivation

The past decade has witnessed an explosion in the number of clinical datasets available to researchers and healthcare professionals. Unfortunately, data are often heterogeneous, scattered, and not uniform among each other in content and format. Extracting useful knowledge from these data is a challenging task. Since these tools should interact with experts from the healthcare domain, it is important that they enable us to explore, explain and summarize the data in a human-friendly way (i.e., in a form that is both concise and easy to understand). This prompts the development of appropriate data mining techniques and tools that can automatically extract relevant information from data and consequently provide insight into various clinical behaviors or processes captured by the data.

For what concerns the clinical domains, it is certainly interesting to show a mining technique that could extract useful rules from any dataset. In fact, a mining algorithm should show its strength in this kind of environment, especially if it could automatically recognize outliers or any other form of error (e.g., bad reported data, inconsistent data, and so on). That is, a mining algorithm should automatically extract rules that hold on most data (approximated rules). An example could be the discovery of a particular dependency between the administration of a drug and a reported adverse reaction to that drug: it is important to discover this kind of rule even if it does not hold for all the patients. An evolution of the previous example could be found when an adverse drug reaction is mainly reported after a certain amount of time or when it is reported after a certain number of previous administrations. In that case, the study of the temporal aspect of the clinical data is key. One of the most meaningful examples of the clinical domain, where the analysis of the approximate and temporal aspect is fundamental, is given by the intensive care domain. In this environment, the medical condition of the patients is continuously monitored, and it is subject to several changes over time. Moreover, due to the critical conditions of many patients, it is necessary to extract insightful information that may hold for most of them (i.e., use approximation). However, the combined analysis of their condition in some aggregate form, such as the variation of a specific vital sign after the administration of a drug, could give medical experts an additional interesting tool. This could be of particular interest if the tool is not only providing rules that regard the current data, but if it is also predicting some specific behavior for the future. As written before, the intensive care domain provides many interesting data to work on. Because of the constant monitoring of the patients, many quantitative data (such as vital signs) are measured and stored periodically, and they are particularly insightful for analyzing trends that they may form.

One way to present knowledge to humans is to use if-then rules, that relate a condition defining a subpopulation of instances (or patients) with observed outcomes. The strength of this relation can be expressed using various statistics, such as support and confidence. This human-friendly form facilitates the exploration, discovery, and possible utilization of these patterns in healthcare. An example could be using a rule mining algorithm to identify a subpopulation of patients that respond better than the others to a particular treatment. If the rule clearly and concisely defines this subpopulation, it can be validated and potentially utilized to improve patient management and outcomes.

There exist many strategies to mine ‘if-then’ rules from the data. The first technique that we find very interesting relies on **Approximate Temporal Functional Dependencies** (AT-FDs). If we consider a plain relational database, functional dependencies are not only a concept for specifying constraints on data and for deriving normal forms [26]. As a matter of fact, some knowledge can be argued by deriving those functional dependencies (FDs) that hold on a given database. As an example, let us consider a simple relational table describing the periodic medical examinations patients undergo in a hospital. That table stores patient personal data such as name, surname, sex, and age, together with symptoms and therapies prescribed at the time of the medical examination. Moreover, a temporal attribute timestamps the medical examination. Typically, patients with the same symptom receive the same therapy. Thus, we can derive a functional dependency between the patient’s symptom and the drug prescribed at the moment of the medical examination.

We may also experience some functional dependencies which hold on “most tuples” of a database, but not on *all* the tuples of that database. For example, we may extract from a clinical database functional dependencies representing knowledge as “most patients with the same symptoms receive therapies of the same type”. We call them *approximate* functional dependencies (A-FDs) [62].

In the clinical domain, the *temporal* aspect should also be considered, and it allows us to define finer constraints. For example, let us consider drugs known to cause a particular *adverse drug reaction*. In this case, further drug prescriptions may follow to mitigate these known effects. For example, suppose that drug d_1 is prescribed to patient p for the treatment of polycythemia, but it causes heartburn. Then, in the next medical examination drug, d_2 would be prescribed to p in order to avoid heartburn. In such a case, prescribed drugs and related adverse reactions determine drugs administered in the next medical examination. We call this dependency a *temporal* functional dependency (T-FD) [115].

For some years, the literature has already considered topics related to approximate functional dependencies [55, 56, 62, 69], and to temporal functional dependencies [32, 57, 110, 113, 115]. To the best of our knowledge, very few studies focused on *approximate temporal* functional dependencies [29, 35, 94]. With AT-FDs, we could combine both the temporal aspect and the approximation of functional dependencies, and that enables us to express concepts such as “most patients with the same symptoms receive therapies of the same type, within a sliding window of 10 days”. In [29], Combi et al. consider the problem of mining approximate T-FDs with different kinds of temporal grouping on clinical data. In [94], Sala extends the concept of approximate T-FD to that of approximate *interval-based* T-FD.

However, the problem of mining (approximate) temporal functional dependencies based on tuple temporal evolution has not been faced yet. The concept of the *temporal evolution* of tuples has been originally introduced by Vianu [110] for the characterization of *Dynamic Functional Dependencies* (DFDs), that allow one to express constraints on tuple evolution in consecutive snapshots of a temporal database.

Furthermore, one notable feature of these dependencies is that comparisons between atemporal attributes are done by using only equality and may represent, for example, features as “For patients undergoing a specific chemotherapy, when the same drug is being prescribed on two consecutive days, the quantity of the drug administered on the latter day depends solely on the drug quantity administered on the former”. However, such dependencies are not well suited for extracting even knowledge when we have to deal with *quantitative* data, such as clinical lab tests, periodic follow-up vital signs, and ICU signals. In particular, it would be important to extract meaningful clinical information, in the form of approximate temporal functional dependencies, where different values for clinical measurements are allowed to have meaningless variations within some specified threshold.

Another strategy to mine ‘if-then’ rules is represented by **pattern mining**, which was introduced by Agrawal [2, 4] for mining market basket data. It gained a lot of popularity in data mining research [49], including clinical data mining [16, 66]. In fact, it is interesting to analyze clinical measures and especially their *temporal* aspect for detecting *temporal patterns*, i.e., time intervals in which one or more time series assume a behavior of interest. Temporal patterns strongly rely on the concept of *temporal abstraction* [98]. Temporal-data abstraction constitutes a central requirement that presently receives much and proper attention. The role of this process is especially crucial in the context of time-oriented clinical monitoring, therapy planning, and exploration of clinical databases. A TA provides a description of a (set of) time series through sequences of temporal intervals that correspond to relevant patterns that are detected in their time courses. Many abstraction mechanisms have been proposed in [15, 27, 93, 98, 101]. These approaches usually deal with data related to only one patient at a time, without the capability to query the whole database of patients. There are many different temporal abstractions, and *trends* represent one of them.

From a qualitative point of view, there are many proposals in literature and some powerful tools that extract qualitative patterns from data ([50, 64, 102]). More recently, methods for assessing temporal relationships between such patterns have been developed ([14, 54, 60]). Despite many differences in their overall approach, all these proposals share common traits in their foundations, as they focus on modeling/extracting *temporal precedence* relationships among patterns. This is expected, as inferring and verifying hypotheses about the cause-effect interaction between events are a core motivation for temporal data mining. However, these methods do not consider the presence of more complex temporal patterns into the dataset. These complex patterns, such as *trends* of a certain vital sign, are very interesting for clinicians to better understand their data. Summing up, it could be interesting to combine these complex patterns. An example could be to discover if some sort of event, such as a drug administration, is capable of changing these trends and how.

When a pattern is focused on a specific target class, we refer to the pattern mining process as to *predictive* pattern (rule) mining [59]. In this case, it considers different rule antecedents (patterns) and consequents (target class).

When predictive pattern mining focuses on the extraction of a set of rules describing important patterns for a specific target class, it is used for knowledge discovery. However, it can also be used to define a classifier [11]. In such a case, predictive patterns can be viewed as nonlinear features helping to improve the overall performance of a classification algorithm. This complementary use of predictive patterns raises interesting problems. One of them could be the possibility to reduce the set of extracted predictive rules with the help of a classification model. In other words, the discovery of rule redundancies that can be eliminated when we combine the rules into a classification model. As a matter of fact, the strength of association rule mining is that it searches the space of rules completely by examining all patterns that frequently occur in the data; however, the number of rules it finds and outputs is often very large. This may hinder the discovery process and the interpretability of the results. Hence, it is desirable to reduce the mined rule set as much as possible while preserving the most important relations (rules, patterns) found in the data. Various rule interestingness statistics and constraints based on such statistics have been proposed to address this problem [44]. However, there are not many studies on the adoption of a classification model to possibly reduce the set of extracted predictive rules.

1.2 Contribution and Overview

The overall goal of this thesis is to study and propose a data mining framework that analyzes clinical data and give medical experts an interesting tool to further validate their data. In particular, we consider the following domains: the intensive care domain given by the MIMIC dataset, the collection of adverse drug reactions given by the Pharmacovigilance domain, and the psychiatric case register. Nevertheless, it is important to underline that the proposed framework is general purpose and may be applied in several other contexts and domains to help different kinds of experts. Figure 1-1 represents an overview of the proposed framework. The idea behind this framework is to collect many different mining techniques that share a common "core". In other words, the core includes the basic notions of widely known concepts, such as *Approximate Temporal Functional Dependencies*, *Predictive Patterns*, or *Temporal Patterns*. For example, in [30] and [96], we proposed two different extensions in the domain of Approximate Temporal Functional Dependency. These extensions inherit all the principles of the known *AT*-FDs (i.e., the *core* of the framework), but they propose new solutions to overcome some of their limitations and to obtain finer rules. This shared core is also useful when we have to create new running prototypes to test the new proposed extensions: in that case, it is not necessary to re-write everything from scratch, but it is possible to reuse all the core functions. In this framework, every additional extension proposed is a *module* that interacts with the core, and the whole framework could mine rules from any given dataset. This part could lead to an interesting scenario: what and how many different rules could we extract from the same dataset? In chapters 4, 5, and 6, we try to answer that question, focusing on intensive care data contained in the MIMIC dataset.

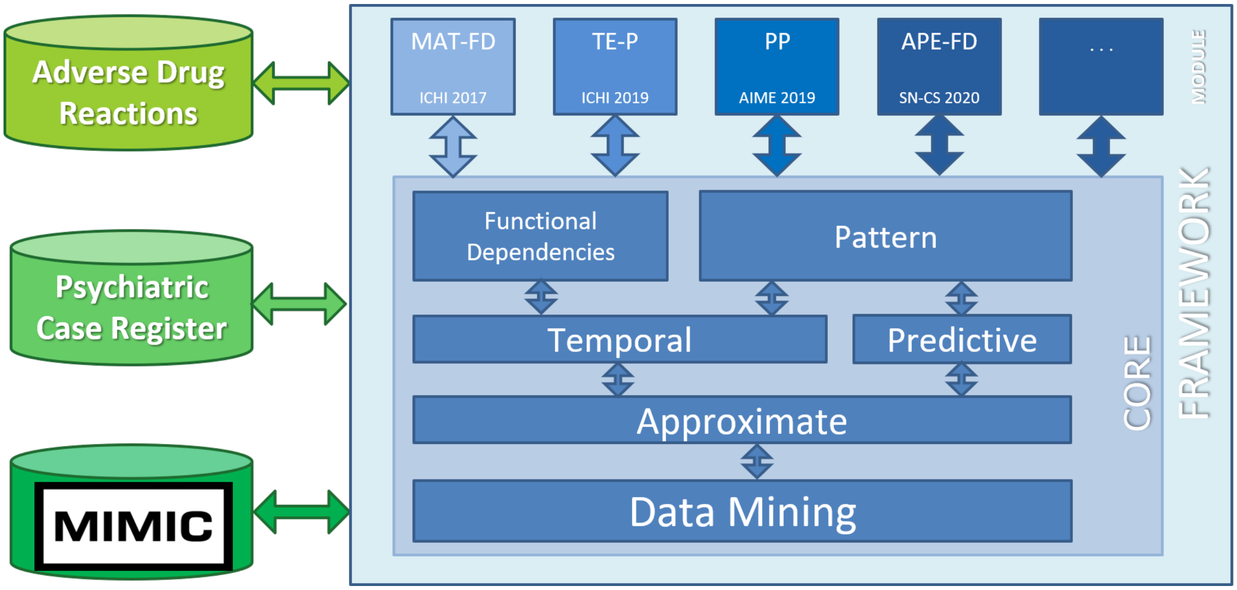


Figure 1-1: Overview of the proposed framework.

In the following, we summarize the contributions of this thesis to obtain the framework described above:

1. **Pure Temporally Evolving Functional Dependencies**

   In the context of functional dependencies, the concept of *temporal evolution* of tuples has been originally introduced by Vianu [110] for the characterization of *Dynamic Functional Dependencies* (DFDs). In this thesis we consider the extensions of DFDs proposed in [32], called Pure Temporally Evolving *T*-FDs (*PE*-FDs) and, in particular, we consider the problem of extracting all Approximate *PE*-FDs, called *APE*-FDs, from a given temporal clinical database.

   Even confirmed by the feedback we had from clinical domain experts, we may say that *APE*-FDs represent dependencies among clinical data in a compact and readable way. Indeed, *APE*-FDs represent knowledge at the schema level. Thus, they may be used as a starting point for deeper data analysis.

   Before moving to the more experimental side of our work, we provide a "negative", yet interesting, result about the complexity of checking *APE*-FDs. Checking a single *APE*-FD against a database instance is *NP*-Complete in the size of the instance (i.e., data complexity), as proved in Appendix A. However, we noticed that the *NP*-completeness of this problem heavily relies on instances that are fictitious and imply properties of data that are unreasonable in many contexts, such as the clinical one. We thus came out with a series of optimizations and heuristics that improves the performances with respect to the more general problem of checking an *APE*-FD against a database instance.

   As we pointed out, mining *APE*-FDs introduces many computational challenges that require techniques inherited from different fields of Computer Science (e.g., model checking and combinatorial optimization). We embedded such techniques in a framework that has been implemented as a running prototype and applied to data from pharmacovigilance and psychiatric domains.

   With respect to the preliminary results presented in [33] and [95], we focus only on *APE*-FDs and do not consider the related temporal association rules. We then propose a new, stronger and more focused definition of *PE*-FD and of the related *APE*-FD, by introducing also a bounded version of temporal evolution of data. Moreover, we provide a detailed discussion and proof of our theoretical results, by introducing a significantly improved and extended presentation with new and more complete examples. Finally, we propose a couple of novel optimization techniques for solving the problem of checking *APE*-FDs.

   In Chapter 3, Section 3.1 formally introduces the concepts of *PE*-FD and *APE*-FD, while Section 3.2 presents some motivating clinical scenarios using *PE*-FDs and *APE*-FDs. Section 3.3 provides a description of the algorithm that checks a single *APE*-FD against a given database plus a series of optimizations and heuristics that may be generally implemented in order to speed-up such verification process. Section 3.4 provides a high-level description of the main features of our prototype for mining such dependencies and the main ideas underlying its implementation; then, it provides interesting mined *APE*-FDs from the psychiatry and pharmacovigilance domains; in the last part of this section, we analyze the performances of the implemented prototype. Section 3.5 draws some conclusions and sketches possible directions for future research. Finally, in the Appendix A we prove the *NP*-Hardness of checking an *APE*-FD against a given temporal database.

The main concepts of this chapter have also been submitted in [96]

Pietro Sala, Carlo Combi, Matteo Mantovani, and Romeo Rizzi. Discovering evolving temporal information: Theory and application to clinical databases. *SN Computer Science*, 1(3):153, May 2020. `doi:10.1007/s42979-020-00160-9`

2. **Multi Approximate Temporal Functional Dependencies**

To remain in the field of functional dependencies, we faced the problem of quantitative analysis combined to the constraint of classic functional dependencies, where the comparison between the values is only by equality. To this end, we propose a new kind of approximate temporal functional dependencies, called Multi Approximate Temporal Functional Dependency (*MAT*-FD), that considers “small variations” for the values of quantitative attributes, while retaining the compactness and the simplicity of *AT*-FDs. To our knowledge, measurements are usually considered, even in an aggregated view to compose trends, at a lower level of abstraction, that is, the level of the specific attribute values such as in the context of temporal association rules [34, 93]. As a matter of fact, there have been few proposals to embed measurements into temporal functional dependencies [116], and nevertheless such proposals focus on specific given trends on attribute values.

Moreover, we designed a mining algorithm for extracting *MAT*-FDs from a given clinical data set. In order to extract knowledge from the data set, the algorithm generates and tests *MAT*-FDs collecting the valid ones, according to some optimality criteria, both for the temporal dimension and for the involved attributes. This means that checking if the data set satisfies a given *MAT*-FDs is an operation that is intensively used during the mining process and its complexity deeply affects the overall complexity of the whole mining process. We propose a graph-based solution for performing such operation and we address its complexity.

Finally, we develop a tool, called SW-MATFDminer, that implements the algorithm above. The tool has been designed to extract *MAT*-FDs regardless of the application domain (i.e., it is a general purpose tool). In particular, we provide some first results from the use of our tool to mine data coming from intensive care units and collected within the MIMIC III database [58].

In Chapter 4, it is introduced the concept of Multi Approximate Temporal Functional Dependencyalong with some examples (Section 4.1). Section 4.2 describes the algorithm used for checking a single *MAT*-FD over a given instance, while

Section 4.3 provides interesting mined *MAT*-FDs from MIMIC III ICU data. Moreover, Section 4.4 draws some conclusions and sketches possible directions for future research.

The main concepts of this chapter have also been published in [30]

Carlo Combi, Matteo Mantovani, and Pietro Sala. Discovering quantitative temporal functional dependencies on clinical data. In *2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, August 23-26, 2017*, pages 248–257. IEEE Computer Society, 2017. URL: `https://ieeexplore.ieee.org/xpl/conhome/8030514/proceeding`, `doi:10.1109/ICHI.2017.80`

3. **Trend-Event Patterns**

There are many systems used for managing and presenting quantitative data, and one of the most widely used is OnLine Analytical Processing (OLAP) system [47, 52, 61, 70]. Such data in the OLAP terminology are called "measures" while qualitative data are called "dimensions". For what concerns the medical domain, OLAP-systems are increasingly adopted since they allow stakeholders to manage, monitor, and analyze information, for example, involving vital signs of the patients (e.g., blood pressures, body temperature, and so on) [101].

Moreover, we may also consider the analysis of the temporal evolution of quantitative data (such as clinical lab tests, periodic follow-up vital signs, ICU signals, and so on), and try to extract useful temporal knowledge.

Considering the current literature and the need of significative synthesis information for several medical domains, we propose a new kind of temporal pattern called *Trend-Event Patterns*, namely *TE*-Ps. The *TE*-P family of temporal patterns focus on the interaction of trends and events. For us a trend is formed by consecutive values of a given measurement attribute that are stationary, increasing, or decreasing under the constraint that all the values of such trend stay within a defined range. In other words, all the values that forms a trend are allowed to have negligible variations within some specified threshold. For instance, *TE*-Ps could express concepts like *"patient's systolic blood pressure was rising before the administration of lisinopril then, after the administration, it stabilized"*.

Another contribution is the development of a tool, called TEPminer, that implements the algorithm to extract *TE*-Ps. Even though TEPminer has been designed to be a general purpose tool (i.e., it is able to mine *TE*-Ps regardless the application domain), we tested it using data coming from intensive care units (ICUs) contained in MIMIC III EHR database [58]. To speed up the mining process, TEPminer exploits multithread functions to analyze data of each patient independently. Finally, we took advantage of OLAP analysis to present some multidimensional analysis on *TE*-Ps, where patterns are evaluated in an aggregate form based on the level of detail we want to use.

In Chapter 5, we propose a new kind of temporal pattern, namely trend-event pattern (*TE*-P) (Section 5.1), then we describe the mining process for *TE*-Ps (Section 5.2). Section 5.3 shows an application of such algorithm using data coming from MIMIC-III

EHR database. Lastly, Section 5.4 outlines the possible future developments of this work.

The main concepts of this chapter have also been published in [75]

Matteo Mantovani, Carlo Combi, and Matteo Zeggiotti. Discovering and analyzing trend-event patterns on clinical data. In *2019 IEEE International Conference on Healthcare Informatics, ICHI 2019, Xi'an, China, June 10-13, 2019*, pages 1–10. IEEE, 2019. URL: `https://ieeexplore.ieee.org/xpl/conhome/8895688/proceeding`, `doi:10.1109/ICHI.2019.8904774`

4. **Predictive Patterns**

   The objective of this work is to study new ways of improving association rule mining that can lead to a smaller set of rules, that are sufficient to capture the essential underlying patterns in the data. This requires analyzing relations among the mined rules and defining criteria for assessing the importance of individual rules w.r.t. other rules. The key principle studied and applied in this work for filtering the rules is rule redundancy. Our approach builds upon the minimum predictive pattern mining idea proposed by Batal and Hauskrecht [11] to eliminate spurious and highly redundant rules, and attempts to improve it by reducing the set of mined minimum predictive rules using an auxiliary classification model that combines the rules into one model. Since in general the search for the optimal set of rules is equivalent to the optimal subset selection problem [63], we propose and experiment with a more efficient greedy rule selection algorithm that avoids the need to explore and evaluate all possible rules subsets.

   We have tested our method on data from MIMIC-III [58] EHR database. More specifically, our goal is to discover patterns that are associated with sepsis and its treatments. We compare our method to the original one [11] and show that the number of rules found by our method is significantly smaller than the original set. Moreover, we show that the performance of the classification model that is based upon our rule set is close or better than classification models built by Batal's rule sets.

   In Chapter 6, we show the problem to identify a small set of predictive patterns. To this end, we recall the concept of *Minimum Predictive Patterns* proposed by Batal and Hauskrecht [11] in Section 6.1.3. In Section 6.1.4 it is shown how to combine predictive patterns using a classification model and in Section 6.1.5 it is illustrated our proposal. Section 6.2 focuses on the experimental part of such approach, showing some interesting results that have been further discuted in Section 6.3. The last Section (6.4) draws some conclusions about such proposal.

   The main concepts of this chapter have also been published in [74]

   Matteo Mantovani, Carlo Combi, and Milos Hauskrecht. Mining compact predictive pattern sets using classification model. In David Riaño, Szymon Wilk, and Annette ten Teije, editors, *Artificial Intelligence in Medicine - 17th Conference on Artificial Intelligence in Medicine, AIME 2019, Poznan, Poland, June 26-29, 2019, Proceedings*,

volume 11526 of *Lecture Notes in Computer Science*, pages 386–396. Springer, 2019. doi:10.1007/978-3-030-21642-9_49

Chapter 2

Background and Related Work

At the beginning of this chapter, we describe some data mining techniques. In particular, the first section recalls basic notions about FUNCTIONAL DEPENDENCY (*FDs*) and its temporal extension, called TEMPORAL FUNCTIONAL DEPENDENCY. Then we introduce the concept of approximation applied to *FDs* and *T*-FDs: APPROXIMATE FUNCTIONAL DEPENDENCY and APPROXIMATE TEMPORAL FUNCTIONAL DEPENDENCY, respectively. Moreover, we illustrate the basic concepts of pattern mining that will be used both for our *Trend-Event Patterns* and *Predictive Patterns* presented in the following chapters. An extension of pattern mining is given by *frequent pattern mining* and its *Apriori* approach. This will lead to the concept of *Predictive Pattern Mining*. Finally, we describe the *Support Vector Machine* (SVM), a classification model we used for predicting patterns.

2.1 Data Mining Techniques

This section briefly recalls the definition of functional dependency (*FD*), and then introduces some extensions of it: temporal functional dependency (*T*-FD) and approximate functional dependency (*A*-FD). Such concepts will lead to the definition of approximate temporal functional dependency (*AT*-FD). For a better understanding of how *FD*, *T*-FD, *A*-FD, and *AT*-FD are related consider Figure 2-1, which graphically depicts the IS A relationships among such dependencies.

2.1.1 Functional Dependencies

The concept of functional dependency (*FD*) comes from the database theory and is defined as follows [26]:

Definition 1 (*Functional Dependency*)**.** Let r be a relationship over the relational schema R: let $X, Y \subseteq R$ be attributes of R. We assert that r fulfills the functional dependency $X \to Y$ (written as $r \vDash X \to Y$) if the following condition holds:

$$\forall t, t' \in r(t[X] = t'[X] \Rightarrow t[Y] = t'[Y])$$

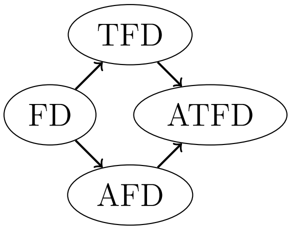


Figure 2-1: A graphical account for the IS_A relationships between Functional dependency (FD), approximate functional dependency (A-FD), temporal functional dependency (T-FD), and approximate temporal functional dependency (AT-FD).

Informally, for all the couples of tuples t and t' showing the same value(s) on X, the corresponding value(s) on Y for those tuples are identical.
With *FDs*, we can express concepts such as *"for each patient with a given symptom the received treatment does not change"*:

$$\textit{Patient, Symptom} \Rightarrow \textit{Treatment.}$$

2.1.2 Temporal Functional Dependencies

Moving closer to the main kind of temporal features we shall consider here, several kinds of temporal functional dependencies (T-FDs) have been proposed in the literature, usually as temporal extensions of the widely known (atemporal) functional dependencies [117]. As an example, we may consider that patients affected by a common pathology p_1 may assume a common therapy t_1 during some month M_1, while in another month M_2 the same patients affected by the same pathology p_1 follow another therapy t_2.

Recently, Combi et al. proposed a framework for T-FDs that subsumes and extends the considered previous proposals [32]. The proposed framework is based on a simple temporal relational data model. This model relies on the notion of *temporal relation*, i.e. a relation extended with a timestamping temporal attribute called *valid time* and denoted by VT. In the temporal relation the valid time represents the temporal dimension, i.e. the time when the fact is true in the modeled world [31].

According to this framework, T-FDs could be expressed using the following syntax: $[E\text{-}Exp(R), t\text{-}Group]X \rightarrow Y$, where $E\text{-}Exp(R)$ is a relational expression on a relational schema R, called *evolution expression.* This evolution expression is based on two temporal views that allow to join tuples that satisfy a specific temporal condition, in order to represent relevant cases of (temporal) evolution. These views are:

- *t-Group* that is a mapping $\mathbb{N} \rightarrow 2^{\mathbb{N}}$, called *temporal grouping.*
- $X \rightarrow Y$ that is a functional dependency on the (atemporal) attributes of $E\text{-}Exp(R)$.

As for the semantics, similarly to the case of standard *FD*s, a T-FD is a statement about admissible temporal relations on a temporal relation schema R with attributes

$U \cup \{VT\}$. A temporal relation $\mathbf{r}$ on the temporal relation schema R satisfies a T-FD $[E\text{-}Exp(R), t\text{-}Group]X \rightarrow Y$ if it is not possible that the relation obtained from $\mathbf{r}$ by applying the expression $E\text{-}Exp(R)$ features two tuples t, t' such that (i) $t[X] = t'[X]$, (ii) $t[VT]$ and $t'[VT]$ belong to the same temporal group, according to the mapping *t-Group*, and (iii) $t[Y] \neq t'[Y]$. In other words, FD $X \rightarrow Y$ must be satisfied by each relation obtained from the evolution relation by selecting those tuples whose valid times belong to the same temporal group.

Temporal grouping enable us to group tuples together over a set of temporal granules, based the `VT` temporal dimension. Four different classes of T-FD have been identified in [32]:

- *Pure temporally grouping T-FD*: $E\text{-}Exp(R)$ returns the original temporal relation $\mathbf{r}$. These rules force *FD* $X \rightarrow Y$, where $X, Y \subseteq U$, to hold over all the maximal sets which include all the tuples whose `VT` belongs to the same temporal grouping;
- *Pure temporally evolving T-FD*: $E\text{-}Exp(R)$ specifies how to derive, through the previously introduced temporal views, the tuples modeling the evolution of objects. No temporal grouping exists, i.e., all the tuples of $\mathbf{r}$ are considered together in one set;
- *Temporally mixed T-FD*: the expression $E\text{-}Exp(R)$ collects all the tuples modeling the evolution of the object. The temporal grouping is applied to the set of tuples generated by $E\text{-}Exp(R)$;
- *Temporally hybrid T-FDs*: the evolution expression $E\text{-}Exp(R)$ selects those tuples of the given temporal relation that contribute to model the evolution of real-world objects (i.e., it removes isolated tuples); then, temporal grouping is applied to the resulting set of tuples.

For example, T-FDs with *Pure Temporally Grouping* are the ones where *t-Group* consists of *granularity* (*Gran*) or *sliding window* (*SW*) *grouping* as follows:

- Grouping on granules (granularity grouping, or *Gran* grouping). A temporal *granularity* is a partition of a temporal domain in indivisible non-overlapping groups, i.e., granules, of time points: minutes, hours, days, months, years as well as working days are *granularities* [28].

  Definition 2 (*Grouping by Gran(i)*)**.** Two tuples $t_1, t_2 \in r$ belong to the same temporal group $Gran(i)$ iff $t_1[VT], t_2[VT] \in Gran(i)$ where $Gran(i)$ is the i^{th} granule of granularity *Gran*.

- Grouping on sliding windows (SW). A sliding window[1] $SW(i, k)$ includes all the time points in interval $[i \ldots i + k - 1]$. Thus, once we fix the length of the SW over relation r (i.e. k in the example), every SW over r will feature that length, and will - at most - include k elements (if relation r has tuples for all the time points of interval $[i \ldots i + k - 1]$).

[1] Actually a sliding window comes with three parameters: granularity, beginning timestamp, and size as the number of time points inside the window.

Definition 3 (*Grouping by* SW*(i,k)*)**.** Two tuples $t_1, t_2 \in r$ belong to the same sliding window $SW(i,k)$ iff $t_1[VT], t_2[VT] \in [i \dots i+k-1]$.

With T-FDs, we can express concepts such as *"for each patient with a given symptom the received treatment does not change, $\underline{\text{considering a time windows of 10 days}}$"*:

[10 days] Patient, Symptom $\Rightarrow$ *Treatment.*

2.1.3 Approximate Functional Dependencies

When considering functional dependencies as a way of finding properties of data instead of a way of specifying constraints on data, *FDs* may be extended through the concept of approximate functional dependency (A-FD). In fact, given a relation **r**, instead of specifying some *FD* as integrity constraints, we might be interested in verifying whether a given *FD* holds on *most* tuples of **r**. Thus, we may allow *some* tuples, for which that *FD* does *not* hold. Therefore, we can define measurements which relate to the error we make in considering an *FD* to hold on **r**. Three different measurements have been defined in [62]. The first one is known as G_1 and considers the number of violating couples of tuples. Formally:

$$G_1(X \to Y, r) = |\{(t, t') \ : \ t, t' \in r \wedge t[X] = t'[X] \wedge t[Y] \neq t'[Y]\}|$$

The related *scaled measurement* g_1 is defined as follows:

$$g_1(X \to Y, r) = G_1(X \to Y, r)/|r|^2$$

where $|r|$ is the cardinality of the relation **r**, i.e. the number of tuples belonging to the relation **r**.

The second one is known as G_2 and considers the number of tuples which violate the functional dependency. Formally:

$$G_2(X \to Y, r) = |\{t \ : \ t \in r \wedge \exists t'(t' \in r \wedge t[X] = t'[X] \wedge t[Y] \neq t'[Y])\}|$$

The related *scaled measurement* g_2 is defined as follows:

$$g_2(X \to Y, r) = G_2(X \to Y, r)/|r|$$

The third one is known as G_3 and considers the minimum number of tuples in **r** to be *deleted* for the *FD* to hold. Formally:

$$G_3(X \to Y, r) = |r| - max\{|s| \mid s \subseteq r \wedge s \models X \to Y\}$$

The related *scaled measurement* g_3 is defined as follows:

$$g_3(X \to Y, r) = G_3(X \to Y, r)/|r|$$

The definition of A-FDs is based on these measurements.

Definition 4 (*Approximate Functional Dependency*)**.** Let r be a relation over the relational schema R: let $X, Y \subseteq R$ be attributes of R. Relation r fulfills an *Approximate Functional Dependency* $X \overset{\varepsilon}{\to} Y$ (written as $r \vDash X \overset{\varepsilon}{\to} Y$) if $G(X \to Y, r)/|r| \leq \varepsilon$, where ε is the maximum acceptable error defined by the user, and G is one of the previously introduced measurements.

Among the several A-FDs that can be identified over a relation r, the minimal A-FD is of particular interest, as many other A-FDs can then be derived from the minimal one. We thus define the minimal A-FD as follows:

Definition 5 (*Minimal A-FD*)**.** Given an A-FD over r, we define $X \overset{\varepsilon}{\to} Y$ to be minimal for r if $r \vDash X \overset{\varepsilon}{\to} Y$ and $\forall X' \subset X$ we have that $r \nvDash X' \overset{\varepsilon}{\to} Y$.

With A-FDs, we can express concepts such as *"for each patient with a given symptom the received treatment does not* <u>*usually*</u> *change"*:

$$\textit{Patient, Symptom} \overset{\varepsilon}{\to} \textit{Treatment.}$$

In the following, we focus only on error measurement $G3$ for all the A-FDs.

2.1.4 Approximate Temporal Functional Dependencies

The concept of AT-FD is introduced in [29], and it relies on two different kind of temporal grouping, both belonging to the class *Pure Temporally Grouping* explained in subsection 2.1.2.

We define the AT-FD with granularity grouping as:

Definition 6 (*AT-FD with Gran grouping*)**.** Let r be a relation over the relational schema R with attributes $U \cup \{VT\}$: let $X, Y \subseteq U$ be attributes of R. Let $Gran$ be the reference granularity. Relation r fulfills an approximate temporal functional dependency (written as $r \vDash [r, Gran]X \overset{\varepsilon}{\to} Y$) iff $g_3([r, Gran]X \to Y, r) \leq \varepsilon$.

That is, the percentage of tuples in the entire relation r to be *deleted* for a AT-FD to hold on all the tuples of r is less than ε; tuples of r are then grouped according to the granule of $Gran$ their VT value belongs to, to evaluate the considered AT-FD. We recall that the count of tuples in r to be deleted refers to the entire relation r, and not to the group - and one tuple may belong to one group only, if we use a $Gran$ grouping.

As for plain A-FD, we can introduce the concept of minimality also for AT-FD.

Definition 7 (*Minimal AT-FD with Gran grouping*)**.** An AT-FD $[r, Gran]X \overset{\varepsilon}{\to} Y$ is said to be minimal for r iff $r \vDash [r, Gran]X \overset{\varepsilon}{\to} Y$ and $\forall X' \subset X$ we have that $r \nvDash [r, Gran]X' \overset{\varepsilon}{\to} Y$.

We define the AT-FD with sliding window (SW) grouping as follows:

Definition 8 (*AT-FD with* SW *grouping*)**.** Let r be a relation over the relational schema R with attributes $U \cup \{VT\}$: let $X, Y \subseteq U$ be attributes of R. Let $\{i \dots i + k - 1\}$ be a sliding window (SW) of length k. The relation r fulfills an approximate temporal functional dependency (written as $r \vDash [r, \{i \dots i+k-1\}]X \overset{\varepsilon}{\to} Y$) iff $g_4([r, \{i \dots i+k-1\}]X \to Y, r) \leq \varepsilon$.

We consider as many SWs as possible, every SW sizing k elements: thus, the first considered sliding window is $i \dots i+k-1$, the second considered sliding window is $i+1 \dots i+k$, the third considered sliding window is $i+2 \dots i+k+1$, and so on. Every SW sets up a group (or *chain*) over which the AT-FD is checked. The AT-FD must hold, with an acceptable amount of error smaller than ε, over the entire database: if we *delete* (as for the measurement g_3) a tuple inside a SW, that tuple will remain *deleted* in *all* the SWs (either preceding or following the current SW) which include that tuple.
Analogously to Definition 7, we can introduce the concept of minimality also for AT-FD with SW grouping.

Definition 9 (*Minimal AT-FD with* SW *grouping*)**.** Given an AT-FD over $[r, \{i \dots i+k-1\}]$, we define $X \overset{\varepsilon}{\rightarrow} Y$ to be minimal for r iff $r \vDash [r, \{i \dots i+k-1\}]X \overset{\varepsilon}{\rightarrow} Y$ and $\forall\ X' \subset X$ we have that $r \nvDash [r, \{i \dots i+k-1\}]X' \overset{\varepsilon}{\rightarrow} Y$.

With AT-FDs, we can express concepts such as *"for each patient with a given symptom the received treatment does not usually change, considering a time windows of 10 days"*:

$$\textit{[10 days] Patient, Symptom} \overset{\varepsilon}{\rightarrow} \textit{Treatment.}$$

An example could be given by Table 2.1, where the attribute `VT` refers to the valid time of the tuple at the *day* granularity; whereas `Duration` refers to the duration of the treatment. If we fix the length of the SW to 5, i.e. every sliding window includes a group (or *chain*) of five days, the first SW will formally include time points {2009-04-11, 2009-04-12, 2009-04-13, 2009-04-14, 2009-04-15}: since relation r in Table 2.1 has tuples for `VT`=2009-04-11 or `VT`=2009-04-14 or `VT`=2009-04-15, the first SW includes 3 tuples having `VT` values 2009-04-11, 2009-04-14, 2009-04-15, respectively. Thus, the following 6 SWs consider all the possible `VT` value groups {2009-04-11, 2009-04-14, 2009-04-15}, {2009-04-14, 2009-04-15}, {2009-04-15}, {2009-04-26, 2009-04-27, 2009-04-28}, {2009-04-27, 2009-04-28}, and {2009-04-28}.

AT-FD $[Contact, \{i \dots i+4\}]$ $Patient \overset{0.4}{\rightarrow} Duration$ holds. Indeed, tuples for which the dependency does not hold, i.e. `Tuple#` 3 and `Tuple#` 6 in the specific example, are those which need to be *deleted* according to the measurement g_3 of Definition 8. More precisely, because the value of attribute `Duration` for `Tuple#` 3 is not 20, and the value for `Tuple#` 6 is not 40, these tuples should be *deleted*. Deleting these two tuples, we obtain a plain T-FD, holding on all the four remaining SWs. The tuples we deleted are less then the 40% of the entire Table 2.1, thus proving that the AT-FD holds even with a threshold ε of 2/6 (i.e. 1/3), which is smaller than 0.4.

If we again consider Table 2.1 and group tuples according to the same six SWs as we did before, we can now check the AT-FD $[Contact, \{i \dots i+4\}]$ $Patient \overset{0.1}{\rightarrow} Treatment$, accepting an error of 10%. The T-FD fails on one tuple (`Tuple#` 5, i.e. 1/6 of the entire relation), which needs to be deleted according to measurement g_3 of Definition 8: thus, the AT-FD does not hold with a ε of 0.1.

Tuple#	VT	Patient	Duration	Treatment
1	2009-04-11	Jackie	20	Atenolol
2	2009-04-14	Jackie	20	Atenolol
3	2009-04-15	Jackie	15	Atenolol
4	2009-04-26	Jackie	40	Lisinopril
5	2009-04-27	Jackie	40	Paracetamol
6	2009-04-28	Jackie	30	Lisinopril

Table 2.1: An example of a table used to show the advantage of approximation in AT-FDs.

2.1.5 Pattern Mining

This subsection recalls the concepts of *pattern*, and its extension, namely *temporal pattern*. An important goal of knowledge discovery is the search for patterns in the data that can help to explain its underlying structure. To be practically useful, the discovered patterns should represent a novelty and in an easy to understand form. Intuitively, a pattern is a behavior or property that we may want to distinguish in the data, or it is used to represent a property in the domain of interest.

Frequent Pattern Mining

Frequent patterns are simply patterns that appear frequently in a dataset. Frequent pattern mining was first introduced by Agrawal [2] for mining market basket data that are in transactional form. The goal was to analyze customer buying habits by finding associations between items that customers frequently buy together. For example, if customers buy bread, it is also likely they buy milk on the same trip to the supermarket. In this example, bread and milk are called items, and the customer's trip to the supermarket is called a transaction. All the transactions are stored in a database.

Frequent patterns can take a variety of forms such as:

1. Itemset patterns: Represent set of items [2, 11, 25, 71, 120].
2. Time interval patterns: Represent temporal relations among states with time durations [10, 12, 53, 78, 85, 118].
3. Sequential patterns: Represent temporal order among items [86, 124, 105, 111].
4. Graph patterns: Represent structured and semi-structured data such as chemical compounds [65, 109, 121].

Frequent pattern mining plays an essential role for discovering interesting regularities that hold in data. Furthermore, frequent pattern mining has been widely used to support other data mining tasks, such as clustering [1, 13] and classification [11, 12, 25, 39, 112].

In the following, a formal definitions of the common concepts of this section:

Definition 10 (*Items*)**.** Let $\Sigma = \{i_1, i_2, \ldots, i_n\}$ denote a set of n attributes, called *items.* Σ is also called the *alphabet.*

Definition 11 (*Database*)**.** Let $D = \{t_1, t_2, \ldots, t_n\}$ be a set of n *transactions*, called *database.* Each transaction $t \in D$ is unique (i.e., has a unique transaction ID), and contains a subset of items in Σ.

Definition 12 (*Itemset pattern*)**.** An itemset pattern is a conjunction of items: $P = i_{q1}\wedge, \ldots, \wedge i_{qk}$ where $i_{qj} \in \Sigma$. If a pattern contains k items, we call it a *k-pattern* (an item is a 1-pattern).

Assume an item $I = (fea, val)$, where fea is a feature and val is a value. Given a data instance $\boldsymbol{x}$, we say that $I \in \boldsymbol{x}$ if $fea(\boldsymbol{x}) = val$ and that $P \in \boldsymbol{x}$ if $\forall I_j \in P : I_j \in \boldsymbol{x}$. Given a dataset $D = \{\boldsymbol{x_i}\}_{i=1}^{n}$, the instances that contain pattern P define a group $D_P = \{\boldsymbol{x_j} | P \in \boldsymbol{x_j}\}$.

Definition 13 (*Subpattern*)**.** We say that pattern *P* is a subpattern of pattern *P'* (*P'* is a superpattern of *P*), denoted as $P \subset$ P', if every item in *P* is contained in *P'*. Note that the empty pattern Φ defines the entire population.

If P is a subpattern of P' ($P \subset P'$), then D_P is a supergroup of D'_P ($D_P \supseteq D'_P$).

Definition 14 (*Support*)**.** The support of pattern *P* in database *D*, denoted as *sup(P,D)*, is the number of transactions t in *D* that contains *P*.

$$supp(P) = \frac{\{t \in D; P \subseteq t\}}{|D|} = \frac{|D_P|}{|D|}$$

Definition 15 (*Frequent Pattern*)**.** Given a user specified minimum support threshold σ, we say that P is frequent pattern if *sup(P,D)* $\geq \sigma$.

For example, consider the transaction data in Table 2.2, where the alphabet of items is $\Sigma = A, B, C, D, E$ and there are 5 transactions (each one of them represents a customer visit to the supermarket). We can see that pattern $P = A \wedge C$ appears in transactions T_1, T_2 and T_4, hence the support of P is 3. If we set the minimum support $sigma = 2$, then the frequent patterns are: A, C, D, E, A $\wedge$ C, A $\wedge$ D.

Transaction	**List of items**
T1	A, C, D
T2	A, B, C
T3	A, D, E
T4	A, C
T5	E

Table 2.2: An example of transactions.

Age	**Sex**	**Heart rate**
Young (< 18)	Male	High (> 100 bpm)
Middle age (18-60)	Male	Medium (60-100 bpm)
Middle age (18-60)	Female	Medium (60-100 bpm)
Senior (> 60)	Female	Low (< 60bpm)

Table 2.3: An example of discretization of numerical values.

Transaction	**List of items**
T_1	Age=Young (< 18), Sex=Male, Heart rate=High (> 100 bpm)
T_2	Age=Middle age (18-60), Sex=Male, Heart rate=Medium (60-100 bpm)
T_3	Age=Middle age (18-60), Sex=Female, Heart rate=Medium (60-100 bpm)
T_4	Age=Senior (> 60), Sex=Female, Heart rate=Low (< 60bpm)

Table 2.4: Transaction format of data in Table 2.3.

The original idea proposed by Agrawal [2] focused on mining transaction data. However, the same concepts can be applied to relational attribute-value data, where each instance is described by a fixed number of attributes. Attribute-value data can be converted into an equivalent transaction data if they are discrete (i.e., data that contain only categorical attributes), and this is a recommended procedure [122]. In this case, it is possible to map each *attribute-value* pair to a distinct *item*.

An example is given in Table 2.3, where the attribute *Heart rate* has been converted into three discrete values: *Low*, *Medium* and *High*. Same procedure was applied to attribute *Age*, with other tree discrete values: *Young*, *Middle age*and *Senior*. In Table 2.4 shows the data in Table 2.3 in transaction format. Let us note that the conversion of attribute-value data into transaction data ensures that all transactions have the same number of items (unless the original data contain missing values). After this transformation, we can apply pattern mining algorithms on the equivalent transaction data.

Pattern Mining Algorithms

One of the most important challenges in pattern mining is to reduce the search space, that can easily become very large. The search space of all possible itemset patterns for transaction data is exponential in the number of items. It means that if Σ is the alphabet of items, there are $2^{|\Sigma|}$ possible itemsets (the powerset of Σ). This search space can be represented by a lattice structure with the empty set at the top and the set containing all items at the bottom, as shown in Figure 2-2. For *attribute-value* data, the search space is exponential in the number of attributes. Given d attributes with V possible values for each one of them, there are $(V+1)^d$ valid itemsets. For more complex patterns, such as sequential patterns, time interval patterns or graph patterns, the search space is even larger than the search space for itemsets. Clearly, the naive approach to generate and count all possible patterns is infeasible. Frequent pattern mining algorithms make use of the minimum support threshold to restrict the search space to a hopefully reasonable subspace that can be explored more efficiently.

Apriori

As noted in [4], frequent patterns have an interesting downward closure property: a pattern can be frequent only if all of its subpatterns are frequent. This property is called the *Apriori property* and it belongs to a category of properties called anti-monotone. This means that if a pattern fails to pass a test, all of its superpatterns will fail the same test as well. The *Apriori algorithm* makes use of an iterative level-wise search and applies the Apriori property to prune the space. It first finds all frequent items (*1-patterns*) by scanning the database and keeping only the items that satisfy the minimum support. Then, it performs the following two phases to obtain the frequent k-patterns using the frequent (k-1)-patterns:

1. *Generation of candidates*: it uses the frequent (*k-1*)-patterns to generate candidate k-patterns. Remove any candidate that contains an infrequent (k-1)-subpattern because it is guaranteed not to be frequent according to the Apriori property.

2. *Minimum Support*: it counts the generated candidates and remove the ones that do not satisfy the minimum support.

This process repeats until no more frequent patterns can be found.

In the following, we illustrate the generation of candidates procedure with an example. Assume the algorithm found the following frequent 2-patterns: $\mathrm{F}_2 = \{A \wedge B, A \wedge C, B \wedge C, B \wedge D\}$. One way to generate candidate k-patterns for itemset mining is by joining two (k-1)-patterns if they share the same *k-2* prefix. Following this strategy, we join $A \wedge B$ with $A \wedge C$ to generate candidate $A \wedge B \wedge C$. Similarly, we join $B \wedge C$ with $B \wedge D$ to generate candidate $B \wedge C \wedge D$. However, it is guaranteed that $B \wedge C \wedge D$ is not frequent because it

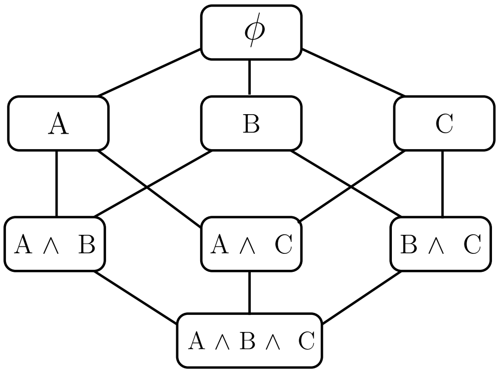


Figure 2-2: Powerset of $\Sigma = A, B, C$.

contains an infrequent 2-subpattern: $C \wedge D \notin F_2$. Therefore, $A \wedge B \wedge C$ is the only candidate that survives the pruning.

Since the Apriori algorithm was proposed, there have been extensive research on improving its efficiency when applied on very large data. These techniques include partitioning [97], sampling [108], and distributed mining [3]. Besides, Apriori has been extended to mine more complex patterns such as sequential patterns [72, 105], graph patterns [65, 109], and time interval patterns [10, 53, 79].

2.1.6 Pattern Mining for Supervised Learning

After the introduction of the main frequent pattern mining algorithms, here we focus on methods that apply pattern mining in the supervised setting, where we have labeled training data of the form $D = \{x_i, y_i\}_{i=1}^n$ (y_i is the class label associated with instance x_i) and we want to mine patterns that can predict the class labels for future instances. In the supervised setting, we are only interested in rules that have the class label in their consequent. Hence, a rule is defined as $P \rightarrow y$, where P (the antecedent) is a pattern and y is a class label. An example of a rule is *Heart Rate = High* $\wedge$ *Temperature = High* $\rightarrow$ *Running = No.*

Concept Learning

One of the most classical method for supervised pattern mining is *concept learning.* Here the learner is presented with training data of the form $D = \{x_i, c(x_i)\}_{i=1}^n$, where $c(x_i)$ is the concept associated with instance x_i. Instances with $c(x_i) = 1$ are called *positive* examples (*members* of the target concept), while the ones with $c(x_i) = 1$ are called *negative* examples (*nonmembers* of the target concept). Let a hypothesis h denote a Boolean-valued function defined over the input space, and let H denote the space of all possible hypotheses the learner may consider: the goal is to find $h \in H$ such that $\forall x, h(x) = c(x)$. In concept learning, the hypothesis space H is determined by the human designer choice of hypothesis representation. Most commonly, H is restricted to include only conjunction of attribute values. For example, assume the data contain three attributes: heart rate, blood pressure, and temperature. Hypothesis $h =< heartrate =?, bloodpressure = high, temperature = hot >$ means that the target concept is true when the value of *blood pressure* is *high* and the value of *temperature* is *hot*, regardless of the value of heart rate. Note that if we use conjunctive hypothesis space, the definition of a hypothesis becomes equivalent to the definition of an itemset pattern. For example, hypothesis h is exactly the same as pattern $bloodpressure = high \wedge temp = hot$. Hence, *the search space for learning conjunctive description hypotheses is the same as the search space of itemset mining for relational attribute-value data.* A useful structure that is used for concept learning is the *general-to-specific* partial ordering of hypotheses. For example, hypothesis $h_1 =< heartrate =?, bloodpressure = high, temperature =? >$ is more general than $h_2 =< heartrate =?, bloodpressure = high, temperature = hot >$, and this is exactly how *subpatterns* are defined, where pattern h_1 is a subpattern of h_2. The general-to-specific partial ordering is used to organize the search through the hypothesis space. It is important to note that concept learning methods rely on two strong assumptions:

1. The hypothesis space H contains the true target concept:

$$\exists\, h \in\ H : h(x) = c(x) \forall\ x \in\ X$$

2. The training data contain no errors (noise free).

For instance, if the hypothesis space supports only conjunctive description and the true target concept is a disjunction of attribute values, then concept learning will fail to learn the concept.

Decision Trees

Another classical method for building classification models is represented by *decision tree induction*. Each internal node in the tree denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. Many algorithms exist to learn a decision tree, such as ID3 [90], CART [19] and C4.5 [91]. All of these algorithms build the decision tree from the root downward in a greedy fashion. One obvious way to obtain a set of classification rules is to first learn a decision tree, then translate the tree into an equivalent set of rules: one rule is created for each path from the root to a leaf node. That is, each internal node along a given path is added to the rule *antecedent* (with conjunction) and the leaf node becomes the rule *consequent*.

An example of decision tree could be found in Figure 2-3, where we want to predicts the concept *Running*. The corresponding rules are:

- R_1: Heart Rate = Regular $\wedge$ Temperature = Regular $\rightarrow$ Running = Yes
- R_2: Heart Rate = Low $\rightarrow$ Running = No
- R_3: Heart Rate = High $\rightarrow$ Running = No
- R_4: Heart Rate = Regular $\wedge$ Temperature = Low $\rightarrow$ Running = No
- R_5: Heart Rate = Regular $\wedge$ Temperature = High $\rightarrow$ Running = No

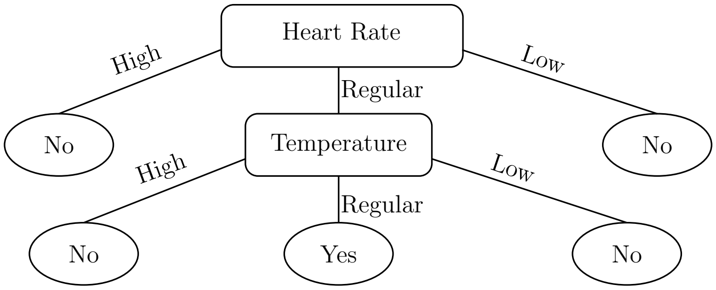


Figure 2-3: An example of decision tree for the concept *Running*.

Because every decision tree induces a partition of the input space, rules that are extracted directly from the tree are *mutually exclusive* and *exhaustive*. Mutually exclusive means that the rules do not overlap (an instance can be covered by only one rule), while exhaustive means that the rules cover the entire input space (every instance is cover by a rule). There are several drawbacks for using rules from a decision tree. First, the extracted rules have a very restrictive form. For example, the attribute of the root note has to appear in every rule. Second, the rules are often difficult to interpret, especially when the original decision tree is large (the rules are often more difficult to interpret than the original tree). Finally, since the decision tree is built greedily, the resulting rules may miss important patterns in the data.

Summing up, concept learning methods search an incomplete hypothesis space because they totally fail when the hypothesis space is complete (the learned concept would exactly replicate the training data). On the other hand, decision tree induction searches the complete hypothesis space (i.e., a space capable of expressing any discrete-valued function). However, the space is searched incompletely using greedy heuristics. In comparison, frequent pattern mining uses a complete hypothesis space and performs a more complete search than decision tree and sequential covering. The reason is that frequent pattern mining examines all patterns that occur frequently in the data instead of relying on greedy choices to explore the patterns.

2.1.7 Temporal Patterns

As we mentioned before, in the clinical domain there are a lot of different data that are collected. Some of them also contain *temporal* information. Let us think, for example, at patients in an intensive care unit: each vital sign is stored with an indication of *when* the vital sign was measured and/or recorded. Temporal data could include time points (e.g., heart rate at 60bpm, at 11:55 a.m., on January 7th, 1987) or temporal intervals. These temporal intervals may be either a part of the original raw input data (e.g., prescription of a drug for seven days), or are abstractions derived from them (e.g., two weeks of fever). A problem could arise when we want to manage time points and intervals from the same input dataset, and in Figure 2-4 we try to illustrate that. In fact, time points and intervals can be intermixed, or the time point series might be sampled or recorded at different frequencies. The sampling process could be set at a fixed time length, as shown in Figure 2-4 for time series (a), which is often the case for automated sampling; or at random periods, as often occurs in manual measurements, as illustrated by time series (b). Moreover, some 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 drug-prescription periods, as shown in series (c), in which the duration of the events is constant, and in series (d), in which the temporal duration is varying. Designing algorithms capable of extracting insightful rules from such data, characterized in various forms, is a challenging topic in temporal data mining research.

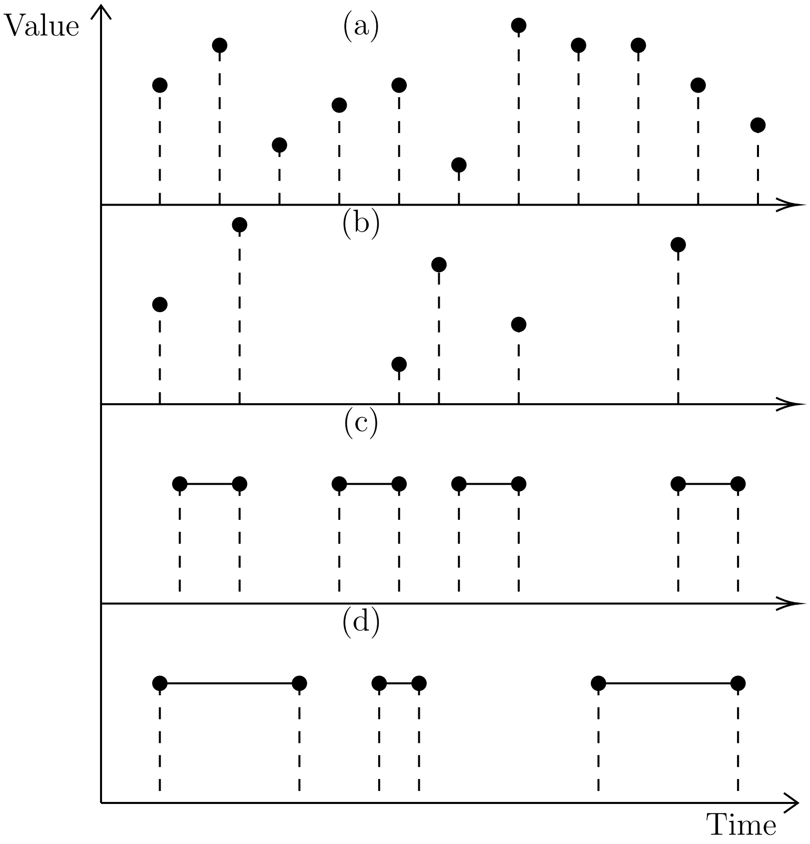


Figure 2-4: Time points (a, b) and intervals (c, d).

Temporal Abstractions

To this extent, it is necessary to introduce the concept of Temporal Abstraction (TA) [98]. The role of this process is especially crucial in the context of time-oriented clinical monitoring, therapy planning, and exploration of clinical databases. A TA provides a description of a (set of) time series through sequences of temporal intervals that correspond to relevant patterns that are detected in their time courses. TAs propose a natural way to define and to describe a representation of temporal data, thus helping to define patterns and to specify temporal relationships between them. The first conceptual model was proposed by Shahar in 1997 [98], with a method called *Knowledge-Based Temporal-Abstraction* (KBTA). Shahar defined a knowledge-based framework, including a formal temporal ontology [98] and a set of computational mechanisms using that ontology [22, 23, 99, 100, 102] specific to the task of creating abstract, interval-based concepts from time-stamped clinical data. The KBTA framework emphasizes the explicit representation of the knowledge required for abstraction of time-oriented clinical data, and facilitates its acquisition, maintenance, reuse, and sharing.

The KBTA theory defines a set of entities, and the most relevant ones for this dissertation are:

- The basic time primitives are *timestamps*, that can be mapped by a time-standardization function into a set of predefined temporal granularity units.

- A time *interval I*, that is an ordered pair of time stamps representing the interval's end points: *[I.start; I.end]*.

- An *event* proposition, that represents the occurrence of an external intentional action or process, such as the administration of a drug.

- *Abstraction Functions*, that transform one or more parameters into an abstract parameter. The "output" abstract parameters can have one of several abstraction types. There are at least three basic abstraction types: *state*, *gradient*, and *rate*.

Since then, many abstraction mechanisms have been proposed in [27, 80, 101]. However, these approaches usually deal with data related to only one patient at a time, without the capability to query the whole database of patients. TAs are also used in [15, 93] to present an approach to pre-process and interpret clinical time series. Their idea is to filter the original time series using temporal abstractions and then to interpret the new and derived time series by both statistical and artificial intelligence methods.

Temporal Trends

One interesting kind of temporal abstraction is represented by trends.

Considering the contributions from the literature on time series and temporal trends, Haimowitz et al. [48] present a temporal pattern-matching system, namely *TrenDx*. *TrenDx* focuses on using efficient general methods to represent and detect predefined temporal patterns in raw time-stamped data. Trend templates describe typical clinical temporal patterns, such as normal growth development, or specific types of patterns known to be associated with functional states or disease states, by representing these patterns as temporal and measurement constraints. The *TrenDx* system has been developed mainly within the domain of pediatric growth monitoring, although examples from other domains have been presented to demonstrate its more general potential.

Wijsen [114] introduces the concept of *trend dependencies*, which allow one to express significant temporal trends, e.g., "salaries of employees should never decrease across months". The temporal dimension is captured by trend dependencies through the concept of time accessibility relation, which can also express time granularities in a simple and elegant way. Trend dependencies can compare attribute values by some comparison operators.

OLAP Analysis of Temporal Patterns

Finally, let us recall that trends and patterns with trends could be analyzed in some aggregate form, through an On-Line Analytical Processing (OLAP) analysis, as shown in [123]. In fact, they can be stored in a multidimensional array structure, called data cube, to be analyzed in an aggregated way. This data cube is composed of many dimensions, where each dimension represents all the possible *qualitative* values described by one or more attributes. The data cube also contains *quantitative* values, and we refer to them as measures. When a dimension is described by more than an attribute, it means that they have a hierarchical relationship. An example of a hierarchy is with the *time* dimension, where attributes such as *week, day, hour, minute, second* are related. The data cube has a multidimensional structure that provides flexibility to analyze data from different perspectives. Consequently, it is possible to query the data cube using OLAP operations, such as drill-down, to navigate from more general data to details; roll-up, to navigate from more specific to more general data; and slice and dice, to define a sub-cube by filtering for one or more dimensions.

2.2 Classification Model - Support Vector Machine

Classifying data is a common task in machine learning. For example, given two classes and a set of data points (where each one of them belongs to one of two classes), the goal is to decide in which class a new data point will be. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a (p-1)-dimensional hyperplane. The concept of *Support Vector Machine* (SVM) was introduced by Cortes and Vapnik in [36]. SVM is a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis. SVM is used when there is a set of training examples, each marked as belonging to one or the other of two categories: here SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model could represent samples as points in space, mapped in a way where samples of the two different categories are divided by a clear gap that is as wide as possible. When new samples are added, SVM maps them into that same space and predicts at what category they belong (i.e., the given category of a new sample is seen by the side of the gap on which they are positioned). There are many hyperplanes that might classify the data. A reasonable choice is to pick the one that represents the largest margin between the two classes. In doing so, the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane, and the linear classifier it defines is known as a maximum-margin classifier.

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors x_i that occur in the dataset. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation $\sum_i \alpha_i k(x_i, x) =$ constant. Note that if $k(x, y)$ becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding database point x_i. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space.

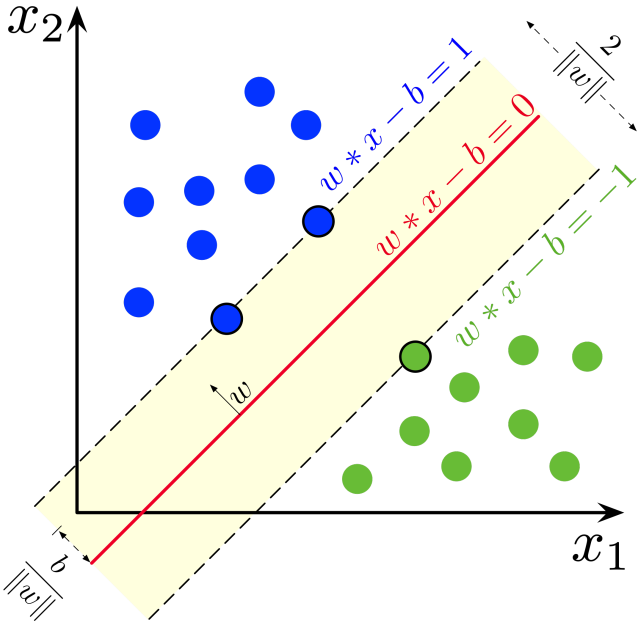


Figure 2-5: In this Figure there are samples from two classes (in blue and green). SVM is trained to discover the maximum-margin hyperplane. Samples on the margin are called the support vectors.

Let us consider a training dataset of n points of the form $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where the y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$\vec{w} \cdot \vec{x} - b = 0$, where $\vec{w}$ is the normal vector to the hyperplane. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$. Figure 2-5 shows an example of the maximum-margin hyperplane.

2.3 Clinical Datasets

In recent years there has been a collective movement for adopting digital health record systems in hospitals. In the US, for example, the number of non-federal acute care hospitals with basic digital systems increased from 9.4 to 75.5% over the 7 year period between 2008 and 2014 [24]. In the following, we present all the clinical datasets that have been used to infer knowledge through techniques described in the following chapters.

2.3.1 MIMIC

MIMIC ('Medical Information Mart for Intensive Care') is a huge database comprising information relating to patients admitted to intensive care units (*ICUs*) of the the Beth Israel Deaconess Medical Center in Boston, Massachusetts [58, 88]. MIMIC includes many data related to patients, such as: vital signs, medications, laboratory measurements, procedure and diagnostic codes, length of stay, survival data, and more. The database supports applications including academic and industrial research, quality improvement initiatives, and higher education coursework.

The MIMIC critical care database is unique and notable for the following reasons:

- it is one of the few freely accessible critical care database of its kind;
- the dataset spans more than a decade, with detailed information about individual patient care;
- analysis is unrestricted once a data use agreement is accepted, enabling clinical research and education around the world.

MIMIC is now at its third major release (called MIMIC-III), it includes data collected between 2001 and 2012, and it is hosted in the PhysioNet system [45]. Data in mimic come from two different data management software used by the Beth Israel Deaconess Medical Center: the original Philips CareVue system, which archived data from 2001 to 2008, was replaced with the new Metavision data management system for the remaining years. In figure 2-6 there is an overview of the underlying structure of MIMIC that is fundamental for the whole data-collection process, while in table 2.5 there are some statistics about MIMIC patient population, as reported in [58]. Figure 2-7 shows the UML schema of MIMIC, and the attributes in each table represent the primary key. Some of the tables have ROW_ID as primary key: this is because, without that attribute, some of the tuples would have been duplicated. For example, in table PRESCRIPTIONS this behavior is expected because each tuple represents an administration of a certain drug, whereas in other tables that behavior could indicate some errors in the data-collection process. Tables without ROW_ID as primary key have one or more attributes that, combined together, return univocally a tuple. In this schema, it is clearly visible that 5 tables are widely interconnected to all the other tables. These tables are:

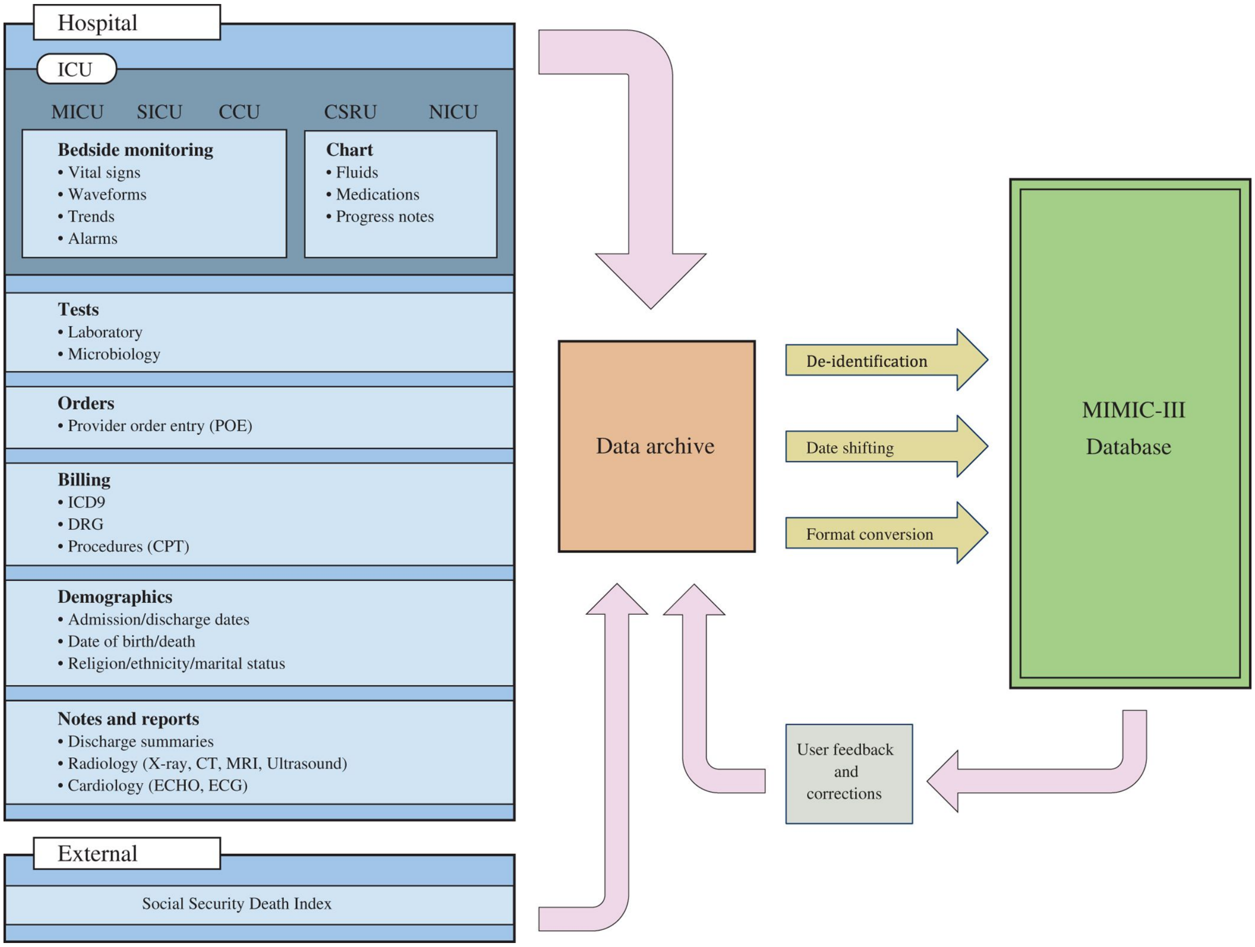


Figure 2-6: Overview of the MIMIC-III critical care system [58].

- Patients - It contains data regarding each patient in the MIMIC dataset, such as: gender, date of birth, and date of death. Let us recall that, because of the de-identification process, these dates do not correspond to the real ones but they preserve their temporal difference.

- Admissions - It gives information regarding a patient's admission to the hospital. Each unique hospital visit for a patient corresponds to an unique Hadm_ID, hence the Admissions table can be considered as a definition table for Hadm_ID. Information available includes data such as temporal data for admission and discharge, demographic information, and the source of the admission.

- ICUstays - It defines a single ICU stay. ICUstay_ID is its generated identifier that is not based on any raw data identifier. The hospital and ICU databases are not intrinsically linked and so do not have any concept of an ICU encounter identifier. Table ICUstays also contains the first and last ICU type in which the patient was cared for: as an ICUstay_ID groups all ICU admissions within 24 hours of each other, it is

Critical care unit	**CCU**	**CSRU**	**MICU**	**SICU**	**TSICU**	**Total**
Distinct patients, no. (% of total admissions)	5,674 (14.7%)	8,091 (20.9%)	13,649 (35.4%)	6,372 (16.5%)	4,811 (12.5%)	38,597 (100%)
Hospital admissions, no. (% of total admissions)	7,258 (14.6%)	9,156 (18.4%)	19,770 (39.7%)	8,110 (16.3%)	5,491 (11.0%)	49,785 (100%)
Distinct ICU stays, no. (% of total admissions)	7,726 (14.5%)	9,854 (18.4%)	21,087 (39.5%)	8,891 (16.6%)	5,865 (11.0%)	53,423 (100%)
Age, years, median (Q1-Q3)	70.1 (58.4–80.5)	67.6 (57.6–76.7)	64.9 (51.7–78.2)	63.6 (51.4–76.5)	59.9 (42.9–75.7)	65.8 (52.8–77.8)
Gender, male, % of unit stays	4,203 (57.9%)	6,000 (65.5%)	10,193 (51.6%)	4,251 (52.4%)	3,336 (60.7%)	27,983 (55.9%)
ICU length of stay, median days (Q1-Q3)	2.2 (1.2–4.1)	2.2 (1.2–4.0)	2.1 (1.2–4.1)	2.3 (1.3–4.9)	2.1 (1.2–4.6)	2.1 (1.2–4.6)
Hospital length of stay, median days (Q1-Q3)	5.8 (3.1–10.0)	7.4 (5.2–11.4)	6.4 (3.7–11.7)	7.9 (4.4–14.2)	7.4 (4.1–13.6)	6.9 (4.1–11.9)
ICU mortality, percent of unit stays	685 (8.9%)	353 (3.6%)	2,222 (10.5%)	813 (9.1%)	492 (8.4%)	4,565 (8.5%)
Hospital mortality, percent of unit stays	817 (11.3%)	424 (4.6%)	2,859 (14.5%)	1,020 (12.6%)	628 (11.4%)	5,748 (11.5%)

Table 2.5: Details of the MIMIC-III patient population by first critical care unit on hospital admission for patients aged 16 years and above. In this table, the critical care units are depicted as follows: CCU is Coronary Care Unit; CSRU is Cardiac Surgery Recovery Unit; MICU is Medical Intensive Care Unit; SICU is Surgical Intensive Care Unit; TSICU is Trauma Surgical Intensive Care Unit.

possible for a patient to be transferred from one type of ICU to another and have the same ICUstay_ID. Moreover it includes the first and last ICU unit in which the patient stayed: note the grouping of physical locations in the hospital database is referred to as ward (Though in practice ICUs are not referred to as wards, the hospital database technically tracks ICUs as "wards with an ICU cost center").

- D_Items - It is the definition table for all the items in the ICU databases (both Metavision and CareVue). The main consequence of having D_Items sourced from these two different databases is that there are duplicate ItemID for each concept. For example, heart rate is captured both as an ItemID of 211 (CareVue) and as an itemid of 220 045 (Metavision). As a result, it is necessary to search for multiple ItemID to capture a single concept across the entire database. Another source of duplicate ItemID is due to the free text nature of data entry in CareVue - as a result there are additional ItemID which correspond to misspellings or synonymous descriptions of a single concept. It is important to search for all possible abbreviations and descriptions of a concept to capture all associated ItemID during the Extract/Transform/Load (ETL) process.

- Caregivers - It provides information regarding the type of caregiver. For example, it would define if a care giver is a research nurse (RN), medical doctor (MD), and so on. This table is also sourced from both CareVue and Metavision, thus imprecisions in the storage process could arise during the usage of these data.

The remaining tables in MIMIC are grouped in 4 categories. Tables that define and track patient stays:

- Callout: Information regarding when a patient was cleared for ICU discharge and when the patient was actually discharged
- Services: The clinical service under which a patient is registered
- Transfers: Patient movement from bed to bed within the hospital, including ICU admission and discharge

Tables that contain data collected in the critical care unit:

- ChartEvents: All charted observations for patients
- DatetimeEvents: All recorded observations which are dates, for example time of dialysis or insertion of lines.
- InputEvents_Cv: Intake for patients monitored using the Philips CareVue system while in the ICU
- InputEvents_Mv: Intake for patients monitored using the iMDSoft Metavision system while in the ICU
- NoteEvents: Deidentified notes, including nursing and physician notes, ECG reports, imaging reports, and discharge summaries.
- OutputEvents: Output information for patients while in the ICU
- ProcedureEvent_Mv: Patient procedures for the subset of patients who were monitored in the ICU using the iMDSoft MetaVision system.

Tables that contain data collected in the hospital record system:

- CPTEvents: Procedures recorded as Current Procedural Terminology (CPT) codes
- Diagnoses_ICD: Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
- DRGCodes: Diagnosis Related Groups (DRG), which are used by the hospital for billing purposes.
- LabEvents: Laboratory measurements for patients both within the hospital and in out patient clinics

- MicrobiologyEvents: Microbiology measurements and sensitivities from the hospital database
- Prescriptions: Medications ordered, and not necessarily administered, for a given patient
- Procedures_ICD: Patient procedures, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system

Tables that are dictionaries:

- D_CPT: High-level dictionary of Current Procedural Terminology (CPT) codes
- D_ICD_Diagnoses: Dictionary of International Statistical Classification of Diseases and Related Health Problems (ICD) codes relating to diagnoses
- D_ICD_Procedures: Dictionary of International Statistical Classification of Diseases and Related Health Problems (ICD) codes relating to procedures
- D_LabItems: Dictionary of ITEMIDs in the laboratory database that relate to laboratory tests

As for the anonymization process, all dates in the database have been shifted to protect patient confidentiality. Dates will be internally consistent for the same patient, but randomly distributed in the future. Dates of birth which occur in the present time are not true dates of birth. Furthermore, dates of birth which occur before the year 1900 occur if the patient is older than 89. In these cases, the patient's age at their first admission has been fixed to 300.

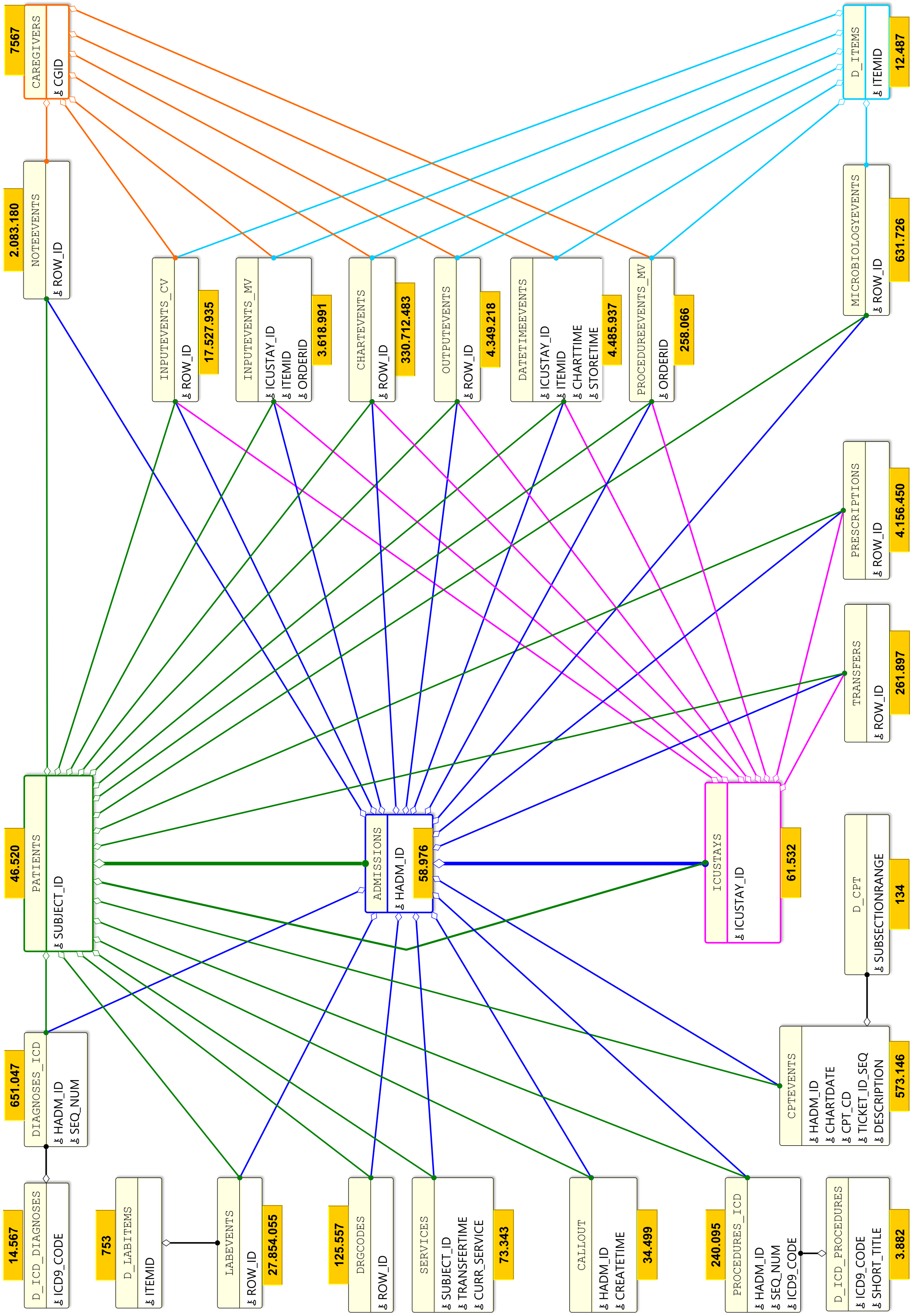


Figure 2-7: Schema of the MIMIC dataset. Each table contains its primary key.

2.3.2 Psychiatric Case Register

Another interesting dataset that we used in our proposed data mining techniques is represented by the temporal clinical data recorded in the Verona Psychiatric Case Register (PCR). Verona is a city in North-East of Italy with about 260 000 inhabitants. The Verona Health District comprises the Verona municipality and other 35 municipalities around the city, for about 460 000 inhabitants. The District is also partitioned in four catchment areas, each one composed by some municipalities. Moreover, biggest municipalities (including Verona itself) are divided in subzones, corresponding to quarters. Each catchment area is served by a Community-based Psychiatric Service (CPS). CPSs aim at providing responses to psychiatric patients' practical as well as psychological and social needs, while trying to alleviate and control their symptoms. Special emphasis is given to integrating different interventions, such as medication, family support, and social work. For this reason permanent staff includes psychiatrists, clinical psychologists, social workers, health visitors, community nurses, ward nurses and counselors. Moreover, CPS structures include

- a psychiatric ward;
- an outpatient department providing psychiatric consultations and individual and family therapy;
- a consultation liaison office that maintains psychiatric integration with other hospital-based medical activities and ensures continuing contact with psychiatric patients when they are hospitalized for medical reasons;
- a 24-hour accident and emergency room;
- a night and week-end emergency room;
- a 24-hour staffed hostel, a group home, and two apartments, offering different levels of supervision.

CPS provides also home visits. These ones can be both in response to emergency calls and, for chronic patients, planned in advance for offering regular, long-term support. Each patient must refer to the CPS leading the area where he lives.

CPS is well integrated, and allows easy and informal access to patients. It is a public service run by the National Health Service. Thus, payment is not required, except for a fee for out-patient visits.

The Verona Psychiatric Case Register (PCR) is an information system collecting information about patients' accesses to CPS since 1979 [7]. At the first contact with the psychiatric service, socio-demographic information, past psychiatric and medical history, and clinical data are routinely collected for patients aged 14 years and over. These data may be updated at successive contacts when required. Each patients' contact with CPS structures is recorded in PCR. Recorded contacts with psychiatrists, psychologists, social workers and psychiatric nurses include:

- attendances at the out-patient clinic;
- domiciliary visits;
- telephone calls;
- day cares provided at the day hospital units;
- all admissions to the acute psychiatric ward and private clinics.

On the other hand, psychiatrists and psychologists in private practice and general practitioners do not report to the PCR.

To each patient a diagnosis is assigned according to ICD-10 categories and then coded into 12 standard diagnostic groups. The International Statistical Classification of Diseases and Related Health Problems 10th Revision (ICD-10) is a coding of diseases and signs, symptoms, abnormal findings, complaints, social circumstances and external causes of injury or diseases, as classified by the World Health Organization (WHO) in 1990 [84]. The diagnosis may be updated at successive contacts if necessary. For all patients who have been registered in the PCR, death and migrations (across or outside the four catchment areas) are recorded by means of annual checks with demographic records of the municipalities of the catchment areas. Data on about 28,700 patients and more than 1,500,000 psychiatric contacts have been recorded in PCR up to now.

Besides patients' personal data (e.g., birth information, health insurance card number, sex, nationality, contacts), patients' medical record, and contact information (including duration, involved operators, motivation, contact kind and conclusions), PCR information system records also education, employment, professional, cohabitants, and marital status of patients.

PCR is used for administrative, clinical, and research purposes. Administrative uses include the analysis of incidence and prevalence rates, number of patients seen, and number of visits made over different time periods for reporting them to local, regional and national level health administrations. Moreover, PCR is used as a basis for calculating direct costs for different groups of patients [6] and for monitoring the effects of changes in resources, organization, and needs.

Clinical purposes include the monitoring of patients who have been in contact with the service for organizing contacts at regular intervals and the provision to clinicians of reports about admissions and contacts for individual patients in a given time period.

Besides their clinical work, psychiatrists and psychologists, along with other staff members, take an active role in research and teaching. Research activities include studying and analysis of the Verona Psychiatric Case Register (PCR). In research activities, PCR is used for:

- epidemiological studies about the utilization of services: for example in [5] Amaddeo et al. investigate the relationship between the lunar cycle and the frequency of patients' contact with CPS;
- studies about the correlation between geographical factors (e.g., position and distance of services) and service utilization [125];

- discovering services-related, area-based, and socio-demographic factors influencing accesses to health services [87, 106, 107];
- studies on mortality among psychiatric patients [46];
- comparisons with other case-register areas [92].

Some of these research studies are based on temporal analysis of the PCR data. Temporal information are contained in several parts of PCR. For example, patients' contacts are temporally qualified with the date in which they occurred, while all personal information about patients (e.g., marital status, employment, and diagnosis) have an associated valid time period. These temporal data can be used for conducting time-trend or time-series studies: for example, epidemiologists may be interested in knowing the number of contacts in different time periods with respect to different factors (e.g., patients' age, diagnosis, and year of birth).

The Psychiatric Case Register follows the schema shown in Figure 2-8. PCR is composed of 7 different tables, that are:

- Patient - Contains all the patients that had at least a contact with the CPS. It contains the main anagraphic data used in the medical records.
- Social - Demographic - Collects all the social - demographic data of a patient and, in particular, all the non-clinical data related to a patient.
- Medical Records - This is the complementary table of Social - Demographic, because it contains all the *clinical* data of a patient.
- Discharge Letter - A patient could be admitted to hospital for period of time that is not initially known. However, when the patient is discharged, the physician writes a discharge letter for the general practitioner. This table contains the letter.
- Prescription - Contains all the drug prescribed by a clinician to a patient.
- Drug - Includes all the data of the drugs prescribed to a patient.
- Physician - Collects data related to physicians. It could be a psychiatrist, a psychologist, or a resident. Usually a patient is assigned to one of them for the whole treatment period.
- Contact - Contains all the data related to therapy sessions with a patient.

From all the tables above, table Contact is the one that has been used intensively in Chapter 3. Because of that, it is worth mentioning the meaning of all of its attributes:

- Contact Num - Records each contact that a physician has with a patient. A contact could be a visit to a patient, a therapy session or a phone call. If the patient is admitted, there could be more daily contacts.

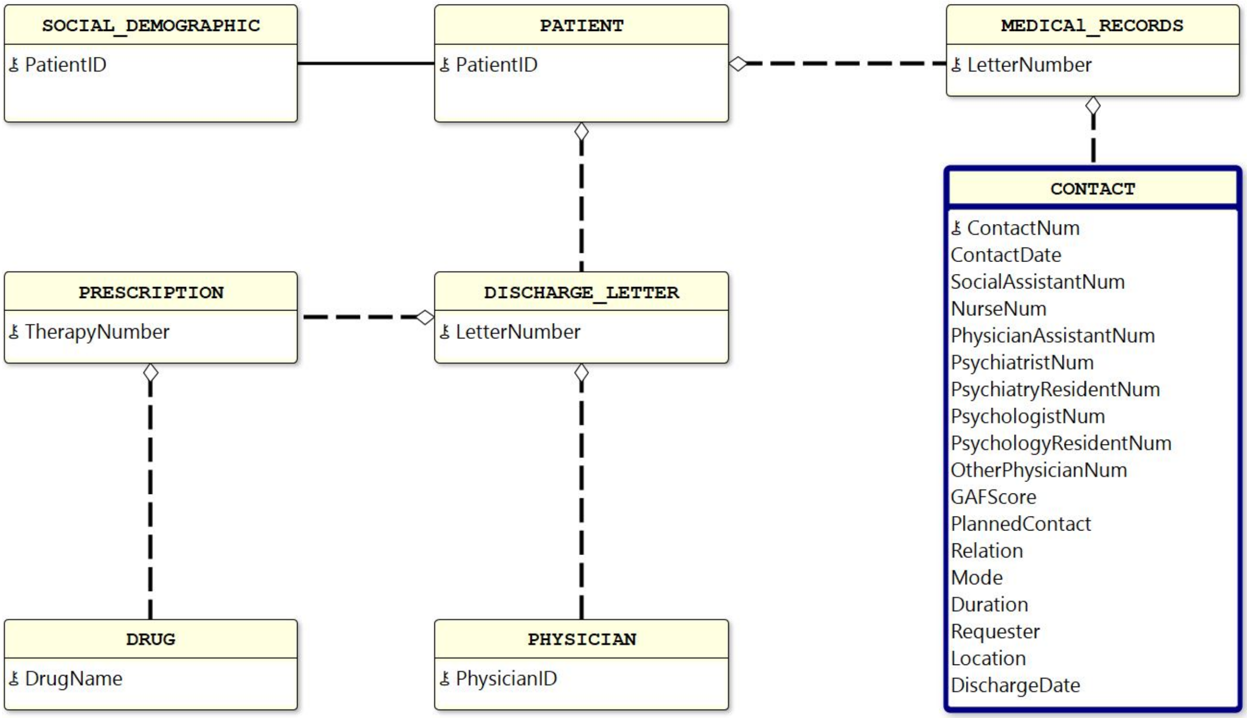


Figure 2-8: Schema of the Psychiatric Case Register dataset. Each table contains its primary key except for table Contact, that is the one used in our analysis.

- Contact Date - The date of a contact.
- Social Assistant Num - The number of social assistants attending a contact.
- Nurse Num - The number of nurses attending a contact.
- Physician Assistant Num - The number of physician assistants attending a contact.
- Psychiatrist Num - The number of psychiatrists attending a contact.
- Psychiatry Resident Num - The number of psychiatry residents attending a contact.
- Psychologist Num - The number of psychologists attending a contact.
- Psychology Resident Num - The number of psychology residents attending a contact.
- Other Physician Num - The number of other physicians attending a contact.
- GAF Score - The Global Assessment of Functioning scale indicates a subjective evaluation of the patient's condition. Scores range from 100, for a patient in perfect condition, to 1. GAF score will be explained with more details below.

- Planned Contact - Boolean value to indicate whether the contact was planned or urgent.
- Relation - A brief textual relation to indicate patients condition and their therapy.
- Mode - Indicates whether the treatment is voluntary or mandatory.
- Duration - The duration of the contact.
- Requester - Indicates who requests the contact: it could be the patient, family, police, general practitioner, or others.
- Location - Where the contact have place.
- Discharge Date - The date when the patient is discharged.

As mentioned above, GAF score is a method to assess patient's condition and it is given by the physician assigned to the patient. GAF score uses a numeric scale from 1 to 100, and the meaning of the score is summarized here:

- **1 to 10** - Persistent inability to maintain minimal personal hygiene, or persistent danger of severely hurting self or others (e.g., recurrent violence) or serious suicidal act with clear expectation of death.
- **11 to 20** - Some danger of hurting self or others (e.g., suicide attempts without clear expectation of death; frequently violent; manic excitement) or occasionally fails to maintain minimal personal hygiene (e.g., smears feces) or gross impairment in communication (e.g., largely incoherent or mute).
- **21 to 30** - Behavior is considerably influenced by serious impairment or delusions or hallucinations, in judgment or communication (e.g., sometimes incoherent, suicidal preoccupation, acts grossly inappropriately) or inability to function in almost all areas (e.g., stays in bed all day, no job, home, or friends)
- **31 to 40** - Some impairment in reality testing or communication (e.g., speech is at times obscure, illogical or irrelevant) or major impairment in several areas, such as work or school, family relations, thinking, judgment or mood (e.g., depressed adult neglects family, avoids friends, and is unable to work; child frequently beats up younger children, is failing at school, and is defiant at home).
- **41 to 50** - Serious symptoms (e.g., suicidal ideation, severe obsessional rituals, frequent shoplifting) or any serious impairment in social, occupational, or school functioning (e.g., no friends, unable to keep a job, cannot work).
- **51 to 60** - Moderate symptoms (e.g., occasional panic attacks, flat affect and circumlocutory speech) or moderate difficulty in occupational, social, or school functioning (e.g., conflicts with peers or co-workers, few friends).

- **61 to 70** - Some mild symptoms (e.g., depressed mood and mild insomnia) or some difficulty in occupational, social, or school functioning (e.g., theft within the household, or occasional truancy), but generally functioning pretty well, has some meaningful interpersonal relationships.

- **71 to 80** - If symptoms are present, they are transient and expectable reactions to psychosocial stressors (e.g., difficulty concentrating after family argument); no more than slight impairment in occupational, social, or school functioning (e.g., temporarily falling behind in schoolwork).

- **81 to 90** - Good functioning in all areas, absent or minimal symptoms (e.g., mild anxiety before an exam), interested and involved in a wide range of activities, generally satisfied with life, socially effective, no more than everyday problems or concerns.

- **91 to 100** - No symptoms. Superior functioning in a wide range of activities, life's problems never seem to get out of hand, is sought out by others because of his or her many positive qualities.

If the score is 0, it means that there are not enough information to assign a score.

2.3.3 Pharmacovigilance

Pharmacovigilance is the science related to the collection, detection, assessment, monitoring, and prevention of suspected adverse reactions induced by drug administrations [119]. Premarketing trials may not detect all adverse reactions induced by the investigated drug due to limitations, such as short duration of the study, and highly selected test population. Adverse Drug Reactions (ADRs) may, indeed, go undetected, and become evident when the drug reaches the market [104]. Therefore, continuous monitoring of the drugs and their effects is needed, even after putting them on sale. The front line of pharmacovigilance consists of signal detection, which aims to identify potential adverse drug reactions that may be unknown. This practice is invaluable, provides early warnings, and requires limited economic and organizational resources [77]. Moreover, there is an advantage in covering every drug on the market and including all categories of patients. There is considerable variation in the use of the term signal [51, 83]. The World Health Organization (WHO) definition, and the most widely cited definition, is “reported information on a possible causal relationship between an adverse event and drug, the relationship being unknown or incompletely documented previously” [41]. When a new adverse event is detected, domain experts must review individual case reports submitted through a spontaneous reporting system for evaluating the signal.

If an investigated signal determines a causal effect between a drug and an adverse reaction, some action must be taken according to the severity of adverse event cases. Sometimes, however, the investigation may assess that the relationships are most likely non-casual. Depending on the nature of the event, a formal study (e.g., epidemiological analysis or large simple clinical trial) may be triggered by the detection of a credible signal.

Thus, possible relationships between one or more adverse reactions, and one or more drugs are investigated. Analysts focus on unknown or completely undocumented relationships. Reports are submitted by physicians, chemists, or citizens. A cause-effect link among ADRs and drugs is suggested. Links can be classified as “suspected” or “concomitant”. Each report includes patient information (such as age, nationality, gender, weight, and outcome of reactions), drug(s) involved in the suspected reaction(s) (identified by their *Anatomical Therapeutic Chemical* - ATC - classification, brand name, dosage), and the description of the occurred adverse reaction(s), the entry date, the period of the adverse reaction, the periods of drug administrations. These temporal data are used to investigate any cause-effect relationship among drugs and reaction(s) in different time periods, or according to the exposure time frame.

Pharmacovigilance is unique among surveillance systems in the range and complexity of medical phenomena under surveillance. Spontaneous reporting systems exist in every country or geographic region, such as the European Union. These systems were created according to different regulations, but they are all founded on a shared base ground. Except for pharmaceutical companies that are legally bound to report suspected ADRs to health authorities, it is usually a voluntary activity by the source reporter (such as health care practitioner, and patient). This is the basis for the term spontaneous reporting. The data elements in individual reports are also subject to considerable qualitative and quantitative deficits in the form of missing or incorrect information and duplicate reporting [81].

In pharmacovigilance, it is important to use common classification systems. To this extent, Anatomical Therapeutic Chemical (ATC) classification system was developed as a drug classification system. ATC is composed by terms that divide active ingredients of drugs into groups based on their chemical, therapeutic characteristics, and of the body part on which they act. It is managed by the World Healthcare Organization Collaborating Centre (WHOCC) for Drug Statistic Methodology and it has been updated periodically since 1976. ATC classes are hierarchically organized in 5 levels:

- First level - Indicates the anatomical main group and consists of one letter (e.g., **B** - Blood and blood forming organs). As shown in Table 2.6, there are 14 main groups.
- Second level - Indicates the therapeutic subgroup and consists of two digits (e.g., B**01** - Antithrombotic agents).
- Third level - Indicates the therapeutic/pharmacological subgroup and consists of one letter (e.g., B01**A** - Antithrombotic agents).
- Fourth level - Indicates the chemical/therapeutic/pharmacological subgroup and consists of one letter (e.g., B01A**C** - Platelet aggregation inhibitors excluding heparin).
- Fifth level - Indicates the chemical substance and consists of two digits (e.g., B01AC**06** - Acetylsalicylic acid)

The fifth level represents a chemical substance in a certain context of use: the same active ingredient may belong to different ATC classes (e.g. the acetylsalicylic acid may be considered as an antiplatelet with code B01AC06 or as an analgesic with code N02BA01). The ATC classification system is a strict hierarchy, meaning that each code necessarily has one and only one parent code, except for the 14 codes at the topmost level which have no parents. The codes are semantic identifiers, meaning they depict in themselves the complete lineage of parenthood.

In the pharmacovigilance environment, the description of the occurred adverse reaction(s) is encoded using the *Medical Dictionary for Regulatory Activities (MedDRA)* classification [76]. MedDRA is a medical terminology used to classify adverse event information associated with the use of pharmaceuticals and other medical products (e.g., medical devices and vaccines). MedDRA, developed by the International Conference on Harmonization (ICH), is continuously enhanced to meet the evolving needs of its users, who include also industry worldwide. The Maintenance and Support Services Organization serves as repository, maintainer, and distributor of MedDRA, as well as the source for the most up-to-date information regarding MedDRA and its application within pharmaceutical industries and regulators. MedDRA subscribers submit proposed changes to the terminology, and this organization (that includes a group of internationally based physicians) reviews all proposed changes and provide a timely response. MedDRA contains a large set of terms, which are structured into five hierarchical levels. The top level System Organ Class (SOC), groups HighLevel Group Terms (HLGT), High-Level Terms (HLT), Preferred Terms (PT), and finally Lowest Level Terms (LLT). At the most specific level, LLT, there are over 70,000 terms which parallel how information

Code	**Contents**
A	Alimentary tract and metabolism
B	Blood and blood forming organs
C	Cardiovascular system
D	Dermatologicals
G	Genito-urinary system and sex hormones
H	Systemic hormonal preparations, excluding sex hormones and insulins
J	Antiinfectives for systemic use
L	Antineoplastic and immunomodulating agents
M	Musculo-skeletal system
N	Nervous system
P	Antiparasitic products, insecticides and repellents
R	Respiratory system
S	Sensory organs
V	Various

Table 2.6: ATC - Anatomical main groups of first level

is communicated. These LLTs reflect how an observation might be reported in practice. Each member of PT defines a single medical concept for a symptom, therapeutic indication, disease diagnosis, investigation, sign, surgical or medical procedure, and medical social or family history characteristic. Each LLT is linked to only one PT, but each PT has at least one LLT (itself) as well as synonyms and lexical variants (e.g., abbreviations, different word order). Related PT are grouped together into HLT based upon pathology, physiology, anatomy, etiology or function, and HLT related to each other are in turn linked to HLGT. Finally, HLGT are grouped into SOC which are groupings by etiology (e.g. Infections and infestations), manifestation site (e.g. Gastrointestinal disorders) or purpose (e.g. Surgical and medical procedures). In addition, there is a SOC to contain issues pertaining to products and one to contain social circumstances. It is important to notice that hierarchy is multiaxial: for example, a PT (Preferred Term) can be grouped in one or more HLT (High Level Term), but it belongs to only one primary SOC (System Organ Class) term. MedDRA is updated twice a year and is available in all european language, chinese, and japanese.

In this thesis we deal with a particular pharmacovigilance dataset, called VigiSegn, that has been created in a previous project in collaboration with the Italian Ministry of Health on drugs surveillance over the Italian territory. The main data used in the analysis of Chapter 3 are summarized in the schema of Figure 2-9. In the following, a brief description of the tables used:

- Patient - Includes data regarding each patient in the VigiSegn dataset, such as sex and age.
- Report - Collects data of the report of an adverse reaction, and in particular all the

relevant dates of such report (i.e., entry date, start and end of the adverse reaction).

- Healthcare Structure - Contains all the data about the healthcare structure that filed the report.
- Drug - Collects multiple combinations of active ingredients that compose the drug (e.g., “acetylsalicylic acid/paracetamol” are the active ingredients that compose Doloflex”).
- Commercial Drug - Contains information about the commercial product based on the reported drug. This table include over 120k medicinal product packages, that are the combination of their commercial name, their quantity and dosage.
- ATC - Includes data regarding the Anatomical Therapeutic Chemical (ATC) classification of the reported drug.
- MedDra - As explained before, Medical Dictionary for Regulatory Activities (MedDRA) is a medical terminology used to classify adverse event information.

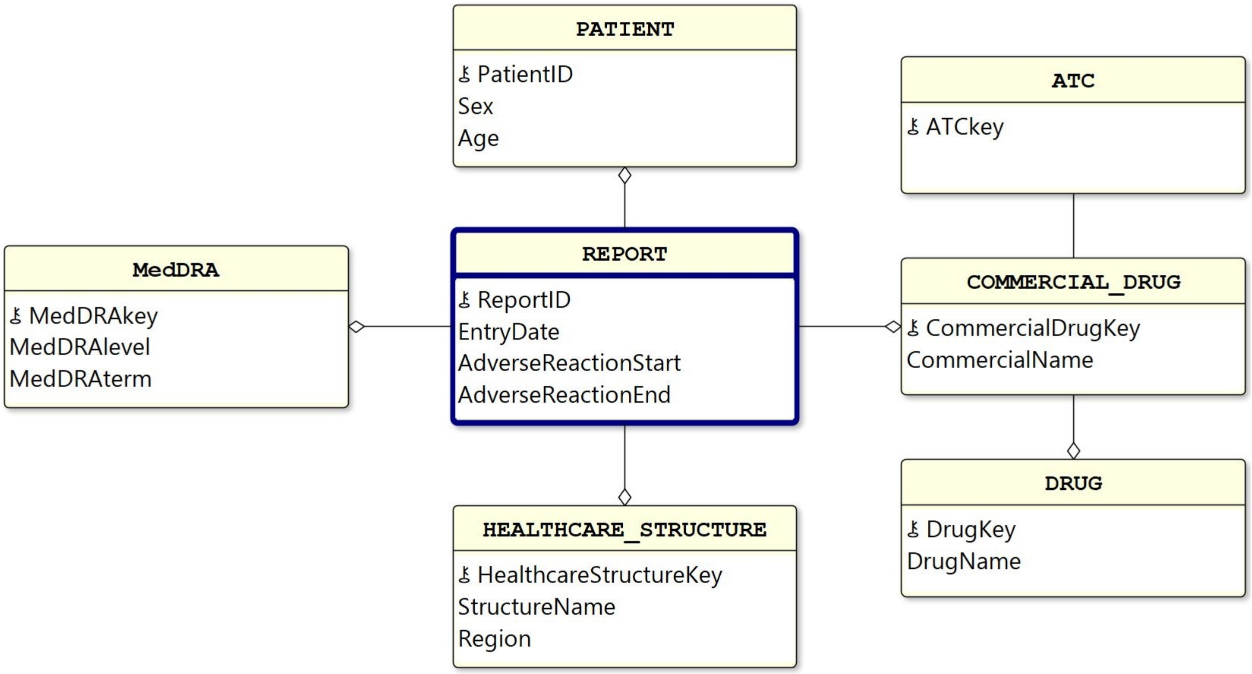


Figure 2-9: Schema of the pharmacovigilance dataset. Each table contains its primary key except for table Contact, that is the one used in our analysis.

Chapter 3

Pure Temporally Evolving Functional Dependencies

In this chapter, we focus on Approximate Temporal Functional Dependency(AT-FDs are defined in Section 2.1), and in particular on the problem of mining AT-FDs based on tuple *temporal evolution* that has not been faced yet. The concept of the *temporal evolution* of tuples has been originally introduced by Vianu [110] for the characterization of *Dynamic Functional Dependencies* (DFDs), that allow one to express constraints on tuple evolution in consecutive snapshots of a temporal database.

Here, we consider the extensions of DFDs proposed in [32], called *Pure Temporally Evolving* T-FDs (PE-FDs) and, in particular, we consider the problem of extracting all Approximate PE-FDs, called APE-FDs, from a given temporal clinical database. *Pure Temporally Evolving* is one particular class of T-FDs, and more details could be found in Section 2.1.2.

Even confirmed by the feedback we had from clinical domain experts, we may say that APE-FDs represent dependencies among clinical data in a compact and readable way. Indeed, APE-FDs represent knowledge at the schema level. Thus, they may be used as a starting point for deeper data analysis. For testing APE-FDs, we implemented a prototype in two real world clinical scenarios: psychiatry and pharmacovigilance (illustrated in Sections 2.3.2 and 2.3.3, respectively). This work has been recently published in [96].

3.1 Discovering Pure Temporally Evolving Functional Dependencies

In the following we focus on *Pure Temporally Evolving Functional Dependencies* (PE-FDs for short), as specified in the framework proposed in [32]. Our temporal functional dependencies will be given on a temporal schema $R = U \cup \{VT\}$ where U is a set of *atemporal attributes* and VT is a special attribute denoting the valid time of each tuple. Hereinafter we assume tuples timestamped with natural numbers (i.e, $\mathcal{D}om(VT) = \mathbb{N}$). Let $J \subseteq U$ be a non-empty subset of U. We define the set W as $W = U \setminus J$ and set $\overline{W}$, which is basically a renaming of attributes in W. Formally, for each attribute $A \in W$, we have $\overline{A} \in \overline{W}$ (i.e., $\overline{W} = \{\overline{A} : A \in W\}$).

Definition 16 (Views *Evolution and Bounded Evolution*)**.** Given an instance $\mathbf{r}$ of R, an instance $\tau_J^{\mathbf{r}}$ of schema $R_{ev} = JW\overline{W}\{VT, \overline{VT}\}$ is defined as follows:

$$\tau_J^{\mathbf{r}} = \left\{ u \;\middle|\; \exists t,t' \left(\begin{array}{c} r(t)\wedge r(t')\wedge t[J]=t'[J]=u[J]\wedge u[W]=t[W]\wedge \\ u[\overline{W}]=t'[W]\wedge t[VT]=u[VT]\wedge t'[VT]=u[\overline{VT}] \\ \wedge t[VT]<t'[VT]\wedge \\ \forall t''((r(t'')\wedge t[VT]<t''[VT])\rightarrow t'[VT]\leq t''[VT]) \end{array} \right) \right\}$$

Schema R_{ev} is called the *evolution schema* of R. We will denote as τ_J^R the view *Evolution* on R that is built by expression $\tau_J^{\mathbf{r}}$ for every instance $\mathbf{r}$ of R. View τ_J^R joins two tuples t_1 and t_2 that agree on the values of the attributes in J (i.e. $t_1[J] = t_2[J]$), if t_2 immediately follows t_1. More precisely, such tuples are joined if $t_1[VT] < t_2[VT]$ and there does not exist a tuple $t \in \mathbf{r}$ with $t[J] = t_1[J]$ and $t_1[VT] < t[VT] < t_2[VT]$ (i.e., there does not exist a tuple that holds at some point in between the valid times of such tuples).

For application purposes, it would be important to consider in an evolution schema only those pairs of consecutive tuples whose the difference between $\overline{VT}$ and VT is within some given bound. Given a parameter $k \in \mathbb{N} \cup \{+\infty\}$, tuples of $\tau_J^{\mathbf{r}}$ are filtered by means of the selection $\Delta_k(\tau_J^{\mathbf{r}}) = \sigma_{\overline{VT}-VT\leq k}(\tau_J^{\mathbf{r}})$ (notice that $\Delta_{+\infty}(\tau_J^{\mathbf{r}}) = \tau_J^{\mathbf{r}}$).

We will denote as $\Delta_k(\tau_J^{\mathbf{r}})$ the view *Bounded Evolution*. It forces to consider only those tuples belonging to $\tau_J^{\mathbf{r}}$ having a temporal distance within the given threshold k. In the following, given a tuple $t \in \tau_J^{\mathbf{r}}$, we denote its temporal distance $t[\overline{VT}] - t[VT]$ with $\Delta(t)$.

Let us now define, by using the introduced temporal view *Evolution*, a slightly restricted version of *Pure Temporally Evolving Functional Dependency* with respect to that defined in [32]. Without loss of generality such definition will allow us to simplify the notation and to focus on a general kind of temporal evolution of considered data.

Definition 17 (Pure Temporally Evolving Functional Dependency)**.** A *Pure Temporally Evolving Functional Dependency* over the temporal schema $R = U \cup \{VT\}$, *PE*-FD for short, is an expression of the form:

$$[\Delta_k(\tau_J^R)]X\overline{Y} \rightarrow \overline{Z}.$$

We have that $X \subseteq W$ and $\overline{Y}, \overline{Z} \subseteq \overline{W}$ with $X \neq \emptyset$ and $|Z| = 1$ (Z contains a single attribute). An instance $\mathbf{r}$ of R fulfills a *PE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \rightarrow \overline{Z}$, written $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \rightarrow \overline{Z}$, if and only if for each pair of tuples $t, t' \in \Delta_k(\tau_J^{\mathbf{r}})$ we have $(t[X] = t'[X] \wedge t[\overline{Y}] = t'[\overline{Y}]) \rightarrow t[\overline{Z}] = t'[\overline{Z}]$.

A *PE*-FD could express dependencies as "A common therapy follows the same symptom for all patients".

Now, we introduce two specializations of *PE*-FDs. Given a *PE*-FD $[\Delta_k(\tau_J^R)]X\ \overline{Y} \rightarrow \overline{Z}$, if set $\overline{Y} = \emptyset$ (i.e, the dependency is $[\Delta_k(\tau_J^R)]X \rightarrow \overline{Z}$), we will say that the *PE*-FD is *simple*. Moreover, if the considered *PE*-FD is of the type $[\Delta_k(\tau_J^R)]X\overline{Y} \rightarrow \overline{X}$ we will say that the *PE*-FD is an *update*. *PE*-FDs featuring both the properties (i.e., *PE*-FDs of type $[\Delta_k(\tau_J^R)]X \rightarrow \overline{X}$) are called *simple updates*. A graphical account of such classes is given in Figure 3-1.

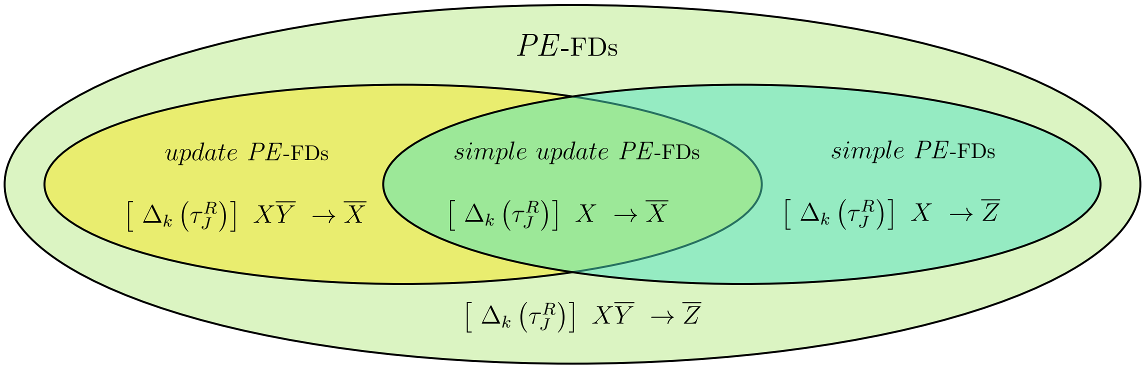

PE-FDs
update PE-FDs
$[\,\Delta_k\,(\tau_J^R)]\ X\overline{Y}\ \to \overline{X}$
simple update PE-FDs
$[\,\Delta_k\,(\tau_J^R)]\ X\ \to \overline{X}$
simple PE-FDs
$[\,\Delta_k\,(\tau_J^R)]\ X\ \to \overline{Z}$
$[\,\Delta_k\,(\tau_J^R)]\ X\overline{Y}\ \to \overline{Z}$

Figure 3-1: A graphical account of how different classes of *PE*-FDs are related.

3.1.1 Approximate Pure Temporally Evolving Functional Dependencies

We add approximation to *PE*-FDs in a very similar way of what has been done for *FDs* in Section 2.1.4.

First, we specialize the measurement G_3, which considers the minimum number of tuples in $\mathbf{r}$ to be deleted for the *FD* to hold, to deal with *PE*-FD as follows:

$$G_3([\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}, \mathbf{r}) = |\mathbf{r}| - \max\{|\mathbf{s}| : \mathbf{s} \subseteq \mathbf{r}, \mathbf{s} \models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}\}$$

By means of G_3 we can define the relative *scaled measurement* g_3 as follows:

$$g_3([\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}, \mathbf{r}) = \frac{G_3([\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}, \mathbf{r})}{|\mathbf{r}|}.$$

Now we are ready to define the *Approximate Pure Temporally Evolving Functional Dependency.*

Definition 18 (Approximate Pure Temporally Evolving Functional Dependency)**.** An *Approximate Pure Temporally Evolving Functional Dependency* over the temporal schema $R = U \cup \{VT\}$, *APE*-FD for short, is an expression of the form:

$$[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$$

with $0 \leq \epsilon \leq 1$, $X \subseteq W$ and $\overline{Y}, \overline{Z} \subseteq \overline{W}$ with $X \neq \emptyset$ and $|Z| = 1$.

An instance $\mathbf{r}$ of R satisfies the *APE*-FD $[\Delta_k(\tau_J^R)]X\,\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$, written $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$, if and only if $g_3([\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}, \mathbf{r}) \leq \epsilon$.

The definitions of simple *PE*-FD and update *PE*-FD may straightforwardly transferred to *APE*-FDs: $[\Delta_k(\tau_J^R)]X \xrightarrow{\varepsilon} \overline{Z}$ is a simple *APE*-FD; $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{X}$ is an update *APE*-FD; $[\Delta_k(\tau_J^R)]X \xrightarrow{\varepsilon} \overline{X}$ is a simple update *APE*-FD.

$\mathbf{r} =$

#	Name	Phys	CT	Duration (minutes)	VT
1	McMurphy	Sayer	self	∼15	1 Jan 2016
2	McMurphy	Sayer	family	∼5	5 Jan 2016
3	McMurphy	Maguire	family	∼15	10 Jan 2016
4	McMurphy	Maguire	self	∼15	15 Jan 2016
5	McMurphy	Maguire	self	∼5	20 Jan 2016
6	McMurphy	Maguire	self	∼5	25 Jan 2016
7	Lowe	Sayer	family	∼15	7 Jan 2016
8	Lowe	Sayer	family	∼15	15 Jan 2016
9	Lowe	Maguire	self	∼15	22 Jan 2016
10	Lowe	Sayer	self	∼5	28 Jan 2016
11	Lowe	Maguire	self	∼15	3 Feb 2016
12	Lowe	Sayer	self	∼5	6 Feb 2016

Table 3.1: An instance **r** of schema *Contact* that stores the phone contacts about two psychiatric cases. Attribute # represents the tuple number and it is used only for referencing tuples in the text (i.e., # does not belong to the schema Contact).

3.2 Some Motivating Clinical Scenarios

In this section we describe and discuss two scenarios, borrowed from the clinical domain, in order to provide examples of how *PE*-FDs and *APE*-FDs work. The first scenario is taken from psychiatric case register. Let us consider the temporal schema $Contact = \{Name, Phys, CT, Dur\} \cup \{VT\}$. Such a schema stores values about a phone-call service provided to psychiatric patients. This service is intended for monitoring and helping psychiatric patients, who are not hospitalized. Whenever a patient feels the need to talk to a physician, she can call the service. Data about calls are collected according to schema *Contact*. For the sake of simplicity, temporal attribute VT identifies the day when the call has been received. In addition, the service may be used by people somehow related to patients, as, for instance, relatives worried about the current condition of a patient.

More precisely, attribute $Name$ identifies patients, $Phys$ identifies physicians, CT (*Contact Type*) specifies the person who is doing the call (e.g., value 'self' stands for the patient himself, 'family' for a relative), and Dur stores information about total duration of calls (value $\sim n$ means approximately n minutes). An instance **r** of R is provided in Table 3.1. Instance $\tau^{\mathbf{r}}_{Name}$, and $\tau^{\mathbf{r}}_{J}$ in general, may be seen as the output of a two-phase procedure. First, table *Contact* is partitioned into subsets of tuples, one for each value of $Name$. Then, each tuple is joined with its immediate successor in its partition, w.r.t. VT values. The whole relation $\tau^{\mathbf{r}}_{Name}$ is provided in Table 3.2. In the following we will use t for referencing tuples of **r** and u for referencing tuples of $\tau^{\mathbf{r}}_{J}$. Moreover, in the following each tuple u in $\tau^{\mathbf{r}}_{J}$ will be identified by the pair of indexes of the tuples in **r** that generate u. For instance, the first tuple of $\tau^{\mathbf{r}}_{J}$ in Figure 3.2 will be denoted by $u_{1,2}$ since it is generated by the join of tuples t_1 and t_2 in **r**.

	#	Name	*Phys*	*CT*	*Dur*	*VT*	$\overline{Phys}$	$\overline{CT}$	$\overline{Dur}$	$\overline{VT}$
$\tau^{\mathbf{r}}_{Name}$	1,2	McMurphy	Sayer	self	∼15	1 Jan 2016	Sayer	family	∼5	5 Jan 2016
	2,3	McMurphy	Sayer	family	∼5	5 Jan 2016	Maguire	family	∼15	10 Jan 2016
	3,4	McMurphy	Maguire	family	∼15	10 Jan 2016	Maguire	self	∼15	15 Jan 2016
	4,5	McMurphy	Maguire	self	∼15	15 Jan 2016	Maguire	self	∼5	20 Jan 2016
	5,6	McMurphy	Maguire	self	∼5	20 Jan 2016	Maguire	self	∼5	25 Jan 2016
	7,8	Lowe	Sayer	family	∼15	7 Jan 2016	Sayer	family	∼15	15 Jan 2016
	8,9	Lowe	Sayer	family	∼15	15 Jan 2016	Maguire	self	∼15	22 Jan 2016
	9,10	Lowe	Maguire	self	∼15	22 Jan 2016	Sayer	self	∼5	28 Jan 2016
	10,11	Lowe	Sayer	self	∼5	28 Jan 2016	Maguire	self	∼15	3 Feb 2016
	11,12	Lowe	Maguire	self	∼15	3 Feb 2016	Sayer	self	∼5	6 Feb 2016

Table 3.2: The evolution expression $\tau^{\mathbf{r}}_{Name}$.

Going back to our example, it is worth noting that tuples t_2 and t_7 are not joined in $\tau^{\mathbf{r}}_{Name}$, even if $t_7[VT] = t_2[VT] + 2$ and there is no tuple t with $t[VT] = t_7[VT] + 1$. This is due to the fact that $t_7[Name] \neq t_2[Name]$ forbids the join in $\tau^{\mathbf{r}}_{Name}$. Moreover, t_1 and t_3 are not joined in $\tau^{\mathbf{r}}_{Name}$. Indeed, the presence of tuple t_2 with $t_1[Name] = t_2[Name] = t_3[Name]$ and $t_1[VT] < t_2[VT] < t_3[VT]$ forbids the join in $\tau^{\mathbf{r}}_{Name}$. Figure 3-2 graphically depicts how pairs of tuples $(t_1, t_2), (t_2, t_3), (t_3, t_4), (t_4, t_5), (t_5, t_6)$ and (t_7, t_8), (t_8, t_9), $(t_9, t_{10}), (t_{10}, t_{11}), (t_{11}, t_{12})$ are joined in $\tau^{\mathbf{r}}_{Name}$ for the two patients, respectively. Basically, each tuple $u \in \tau^{\mathbf{r}}_{Name}$ corresponds to an edge in Figure 3-2 while we have a node for each tuple in $\mathbf{r}$.

Let us now discuss some temporal dependencies we can derive from such data. We could be interested in verifying whether there is some relationship between some previous features of patient's call and the fact that the considered call was either with him or with a relative. In our example, we have that $\mathbf{r} \models [\Delta_5(\tau^{Contact}_{Name})]Phys, \overline{Phys} \rightarrow \overline{CT}$.

In other words, given consecutive calls related to the same patient within 5 days, the couple composed by the physician of the first call and by the physician of the next one determines the type of contact of the next call. And it holds for all patients. However, if we consider a wider time window of 6 days, we have that $\mathbf{r} \not\models [\Delta_6(\tau^{Contact}_{Name})]Phys, \overline{Phys} \rightarrow \overline{CT}$, because of pairs (t_2, t_3) and (t_{10}, t_{11}). More precisely, we have that $t_2[Name] = t_3[Name] =$ *"McMurphy"*, $t_{10}[Name] = t_{11}[Name] =$ *"Lowe"*, $t_2[Phys] = t_{10}[Phys] =$ *"Sleepy"*, $t_3[Phys] = t_{11}[Phys] =$ *"Patel"*, but $t_3[CT] \neq t_{11}[CT]$ (i.e., $t_3[CT] =$ *"family"*, and $t_{11}[CT] =$ *"self"*). In other words, we have that the set of tuples $\{u_{2,3}, u_{10,11}\}$ does not satisfy the *FD* $Phys, \overline{Phys} \rightarrow \overline{CT}$.

The two proposed *PE*-FDs differ only for the maximum temporal distance allowed. In particular, tuple $u_{10,11}$ is one of the responsibles for $\mathbf{r} \not\models [\Delta_6(\tau^{Contact}_{Name})]$ $Phys$, $\overline{Phys} \rightarrow \overline{CT}$, but it does not belong to $\Delta_5(\tau^{Contact}_{Name})$ because $\Delta(u_{10,11}) > 5$. This allows us to point out a general property of *PE*-FDs and of *APE*-FDs too. Given a *PE*-FD $[\Delta_k(\tau^R_J)]X\overline{Y} \rightarrow \overline{Z}$, if we have that for every instance $\mathbf{r}$ of R it holds $\mathbf{r} \models [\Delta_k(\tau^R_J)]X\overline{Y} \rightarrow \overline{Z}$, then for every $h \leq k$ it holds $\mathbf{r} \models [\Delta_h(\tau^R_J)]X\overline{Y} \rightarrow \overline{Z}$.

Moving to the problem of mining approximate dependencies, if we consider *APE*-FD $[\Delta_6(\tau^R_{Name})]Phys, \overline{Phys} \xrightarrow{\varepsilon} \overline{CT}$ with $\epsilon = \frac{1}{12}$, we have that $\mathbf{r} \models [\Delta_6(\tau^R_{Name})]$ $Phys, \overline{Phys} \xrightarrow{\varepsilon} \overline{CT}$.

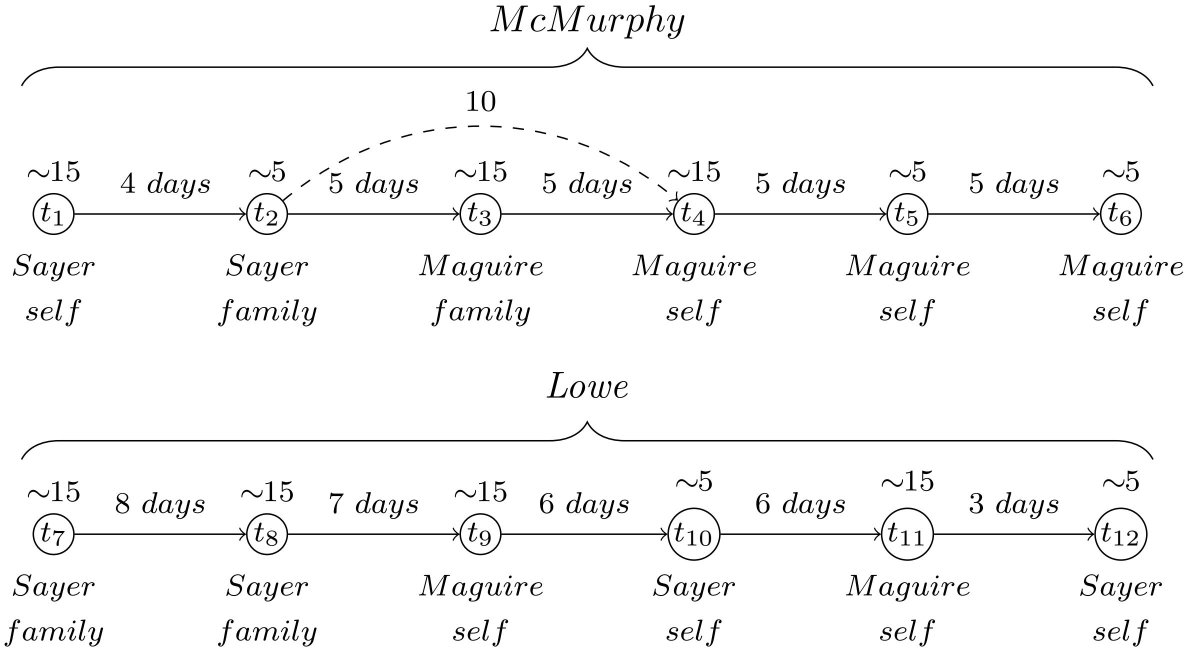


Figure 3-2: A graph-based representation of $\tau^{\mathbf{r}}_{Name}$. Nodes represent tuples and are labeled by the corresponding tuple number. Values for attribute Dur are reported above each node. Values of $Phys$ and CT attributes are reported below every node, respectively. Every edge (t_i, t_j) is labeled by value $\Delta(u_{i,j}) = t_j[VT] - t_i[VT]$ (i.e., the temporal distance between two tuples). The dashed edge represents a different scenario where t_2 and t_4 are joined, if t_3 would be deleted, as explained below for APE-FDs.

Indeed, by considering relation $\mathbf{r}' = \mathbf{r} \setminus \{t_3\}$, this dependency would hold without the need of approximation (i.e., if tuple t_3 is deleted from relation $\mathbf{r}$). More precisely, we have $\tau^{\mathbf{r}'}_{Name} = \tau^{\mathbf{r}}_{Name} \setminus \{u_{2,3}, u_{3,4}\} \cup \{u_{2,4}\}$. Tuple $u_{2,4}$ was not originally in $\tau^{\mathbf{r}}_{Name}$ because of tuple t_3. Figure 3-2 depicts this new scenario, by replacing edges (t_2, t_3) and (t_3, t_4) with the dashed edge (t_2, t_4). Moreover, we have $\mathbf{r'} \models [\Delta_{+\infty}(\tau^{R}_{Name})]Phys, \overline{Phys} \rightarrow \overline{CT}$. Thus, it holds $\mathbf{r} \models [\Delta_{+\infty}(\tau^{R}_{Name})]Phys, \overline{Phys} \xrightarrow{\varepsilon} \overline{CT}$ with $\epsilon = \frac{1}{12}$.

The second example we propose is borrowed from the internal medicine domain. As another simple example of how view τ^{R}_{J} works, let us consider the temporal schema $ThCy = \{PatId, Phys, Dos\} \cup \{VT\}$. Such a schema allows one to store the values about cycles of therapies in which a specific, fixed, drug is administered to a patient by a given physician. Figure 3.3 (a) depicts an instance $\mathbf{r}$ of R. Figure 3.3 (b) shows the result of view τ^{ThCy}_{PatId} to $\mathbf{r}$. It is easy to see that tuples t_1 and t_5 are not joined in $\tau^{\mathbf{r}}_{PatId}$. Even if $t_5[VT] = t_1[VT] + 1$ the fact that $t_1[PatId] \neq t_5[PatId]$ forbids the join in $\tau^{\mathbf{r}}_{PatId}$. Moreover t_1 and t_3 are not joined in $\tau^{\mathbf{r}}_{PatId}$. Even if $t_1[PatId] = t_3[PatId]$, we have that the presence of tuple t_2 with $t_1[PatId] = t_2[PatId] = t_3[PatId]$ and $t_1[VT] < t_2[VT] < t_3[VT]$ forbids the join in $\tau^{\mathbf{r}}_{PatId}$. In Figure 3-3 (a) we have a graphical account of how the pairs of tuples $(t_1, t_2), (t_2, t_3), (t_3, t_4), (t_5, t_6), (t_6, t_7)$ and (t_7, t_8) are joined in $\tau^{\mathbf{r}}_{PatId}$. In both the graphs

(a)

$\mathbf{r} =$

#	**PatId**	**Phys**	**Dos**	**VT**
1	1	Shepherd	30 mg	1 May 2016
2	1	Stevens	60 mg	5 May 2016
3	1	Shepherd	40 mg	20 May 2016
4	1	Shepherd	40 mg	27 May 2016
5	2	Stevens	30 mg	2 May 2016
6	2	Stevens	40 mg	9 May 2016
7	2	Shepherd	60 mg	20 May 2016
8	2	Stevens	20 mg	25 May 2016

(b)

$\tau^{\mathbf{r}}_{PatId} =$

#	**PatId**	**Phys**	**Dos**	**VT**	$\overline{\mathbf{Phys}}$	$\overline{\mathbf{Dos}}$	$\overline{\mathbf{VT}}$
1,2	1	Shepherd	30 mg	1 May 2016	Stevens	60 mg	5 May 2016
2,3	1	Stevens	60 mg	5 May 2016	Shepherd	40 mg	20 May 2016
3,4	1	Shepherd	40 mg	20 May 2016	Shepherd	40 mg	27 May 2016
5,6	2	Stevens	30 mg	2 May 2016	Stevens	40 mg	9 May 2016
6,7	2	Stevens	40 mg	9 May 2016	Shepherd	60 mg	20 May 2016
7,8	2	Shepherd	60 mg	20 May 2016	Stevens	20 mg	25 May 2016

(c)

$\tau^{\mathbf{r}}_{Phys} =$

#	**Phys**	**PatId**	**Dos**	**VT**	$\overline{\mathbf{PatId}}$	$\overline{\mathbf{Dos}}$	$\overline{\mathbf{VT}}$
1,3	Shepherd	1	30 mg	1 May 2016	1	40 mg	20 May 2016
1,7	Shepherd	1	30 mg	1 May 2016	2	60 mg	20 May 2016
3,4	Shepherd	1	40 mg	20 May 2016	1	40 mg	27 May 2016
7,4	Shepherd	2	60 mg	20 May 2016	1	40 mg	27 May 2016
5,2	Stevens	2	30 mg	2 May 2016	1	60 mg	5 May 2016
2,6	Stevens	1	60 mg	5 May 2016	2	40 mg	9 May 2016
6,8	Stevens	2	40 mg	9 May 2016	2	20 mg	25 May 2016

Table 3.3: The instances $\tau^{\mathbf{r}}_{PatId}$ (b) and $\tau^{\mathbf{r}}_{Phys}$ obtained from applying views τ^{ThCy}_{PatId} and τ^{ThCy}_{Phys} to the instance $\mathbf{r}$ (c) respectively.

depicted in Figure 3-3 (a) nodes are labeled with the tuple number and the value for *Dos* attribute is reported above each node. Moreover, we recall from the previous example that each edge (t_i, t_j) is labelled with the value $t_j[VT] - t_i[VT]$ (i.e., the temporal distance between two tuples). In the scenario for $\tau^{\mathbf{r}}_{PatId}$ the value for the attribute *Phys* is reported below each node, while for $\tau^{\mathbf{r}}_{Phys}$ the value of *PatId* is reported below each node.

We would like to point out that it may be the case that a tuple $t \in \mathbf{r}$ has a more than one immediate successor in $\tau^{\mathbf{r}}_{J}$ (i.e. is joined with more than one tuple). It is the case of view $\tau^{\mathbf{r}}_{Phys}$ shown in Figure 3-3 (b), where tuples are joined with respect to the values of attribute *Phys*. We have that, since Dr. *Shepherd* makes two drug administrations at $VT = 20$, tuple t_1 has both tuples t_3 and t_7 as its immediate successors. We will see that the number of immediate successors of a tuple in $\tau^{\mathbf{r}}_{J}$ will play a major role in some of the following complexity results.

In this domain we could be interested in understanding whether there are dependencies among previous and current drug dosages for a given patient, possibly considering the physicians administering the drug.

In the example depicted in Figures 3.3 and 3-3, we have that $\mathbf{r} \models [\Delta_{+\infty}(\tau^{R}_{PatId})]$ $Dos, Phys, \overline{Phys} \rightarrow \overline{Dos}$. It means that the dosage and the couple of physicians related to a drug administration and to the next one respectively, determine the next drug dosage.

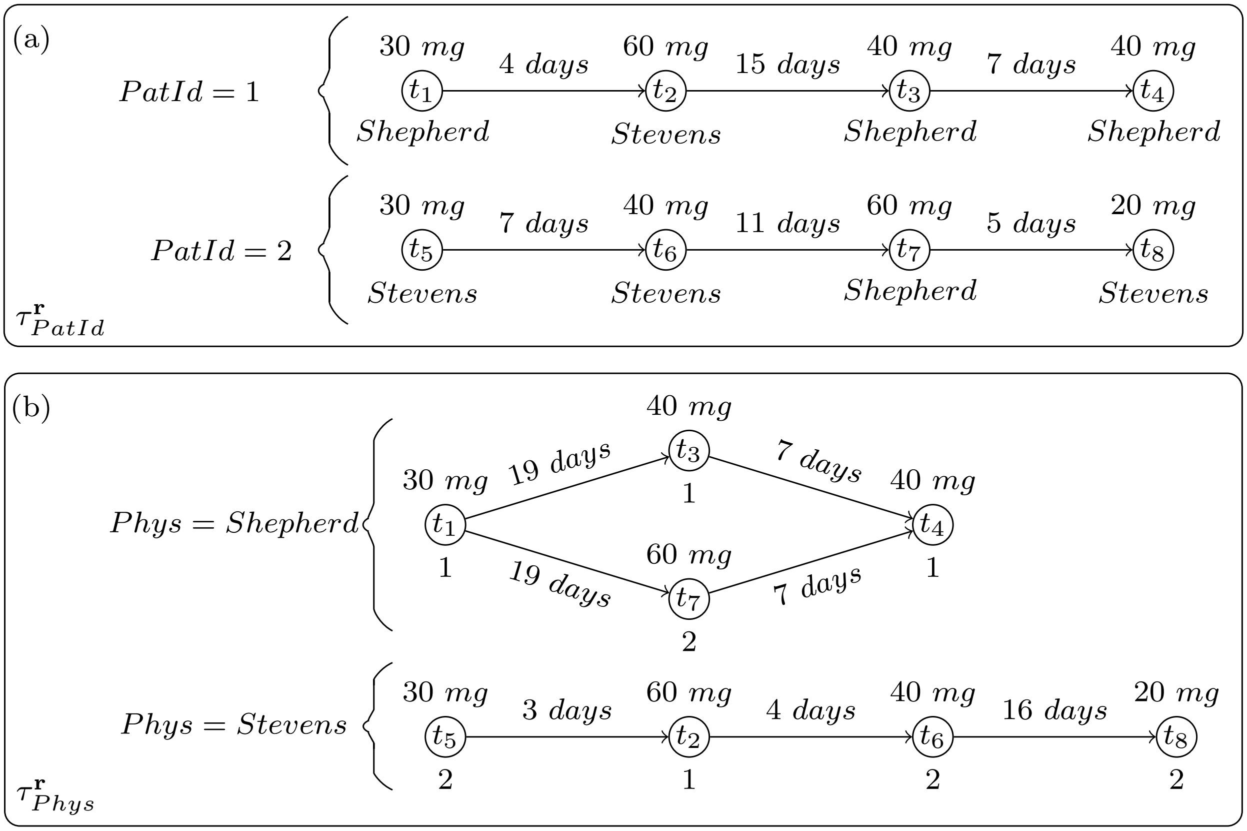


Figure 3-3: A graphical account for the instances $\tau^{\mathbf{r}}_{PatId}$ (a) and $\tau^{\mathbf{r}}_{Phys}$ (b) related to the instance $\mathbf{r}$ shown in Figure 3.3 (a).

However, $\mathbf{r} \not\models [\Delta_{\infty}(\tau^{R}_{PatId})]Phys, \overline{Phys} \rightarrow \overline{Dos}$ because both tuples $u_{1,2}$ and $u_{7,8}$ belong to $\Delta_{\infty}(\tau^{R}_{PatId})$. More precisely, we have that $Phys, \overline{Phys} \rightarrow \overline{Dos}$ does not hold on any instance of $\Delta_{k}(\tau^{R}_{PatId})$ that contains both $u_{1,2}$ and $u_{7,8}$, since $u_{1,2}[Phys] = u_{7,8}[Phys] = Shepherd$ and $u_{1,2}[\overline{Phys}] = u_{7,8}[\overline{Phys}] = Stevens$ but $u_{1,2}[\overline{Dos}] = 60\ mg$ is not equal to $u_{7,8}[\overline{Dos}] = 20\ mg$.

On the other hand, $\mathbf{r} \models [\Delta_{4}(\tau^{R}_{PatId})]Phys, \overline{Phys} \rightarrow \overline{Dos}$ because having 4 as maximum allowed time distance implies $u_{7,8} \notin \Delta_{4}(\tau^{\mathbf{r}}_{PatId})$ (i.e., $t_8[VT] - t_7[VT] > 4$) and thus the conflict between $u_{1,2}$ and $u_{7,8}$ no longer exists. Furthermore, it is easy to see by considering the pairs $u_{1,2}$ and $u_{5,6}$ that $\mathbf{r} \not\models [\Delta_{+\infty}(\tau^{R}_{PatId})]Dos, \overline{Phys} \rightarrow \overline{Dos}$. However, by shrinking the maximum allowed time distance to 6 we obtain $u_{5,6} \notin \Delta_{6}(\tau^{\mathbf{r}}_{PatId})$ and thus we have that $\mathbf{r} \models [\Delta_{6}(\tau^{R}_{PatId})]Dos, \overline{Phys} \rightarrow \overline{Dos}$.

Obviously in an instance $\tau^{\mathbf{r}}_{J}$ there may be more than one pair (u, u') which generates a conflict. Let us consider the simple update *PE*-FD $[\Delta_{+\infty}(\tau^{R}_{PatId})]\ Dos \rightarrow \overline{Dos}$. Such *PE*-FD does not hold on $\mathbf{r}$ (i.e., $\mathbf{r} \not\models [\Delta_{+\infty}(\tau^{R}_{PatId})]Dos \rightarrow \overline{Dos}$). This can be shown using the pair $u_{1,2}$ and $u_{5,6}$ as a witness for a conflict since $u_{1,2}[Dos] = u_{5,6}[Dos] = 30\ mg$ and $u_{1,2}[\overline{Dos}] \neq u_{5,6}[\overline{Dos}]$. However, in this case we have that the conflicting pairs are more than one. As a matter of fact all pairs $(u_{1,2}, u_{5,6})$, $(u_{2,3}, u_{7,8})$, and $(u_{3,4}, u_{4,6})$ are conflict-generating. If we want to rule all the conflicts out by playing on the maximum allowed distance, we have to set it to 6 and then we have $\mathbf{r} \models [\Delta_{6}(\tau^{R}_{PatId})]Dos \rightarrow \overline{Dos}$.

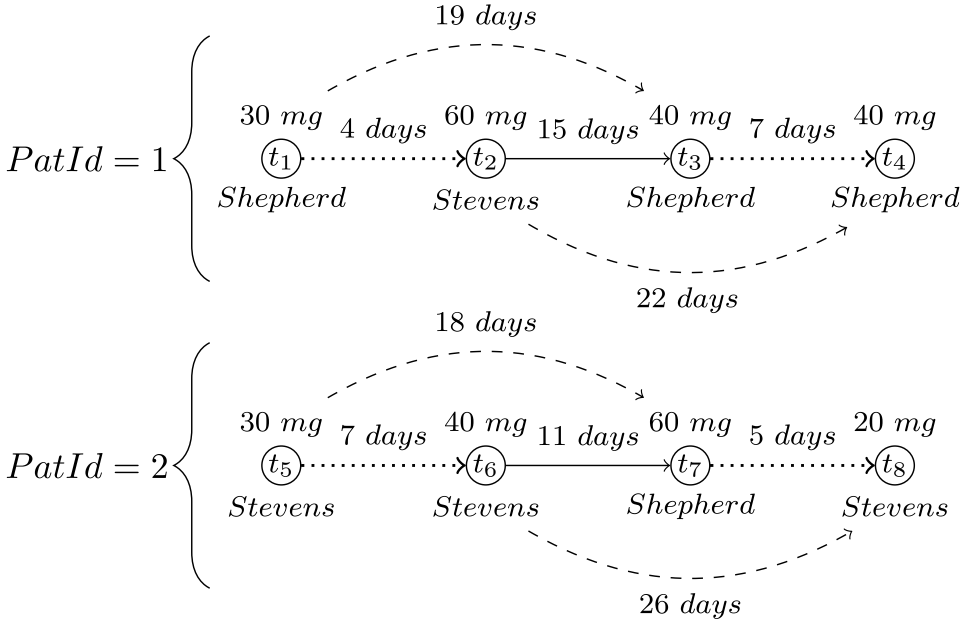


Figure 3-4: A graphical account for the possible changes on the view $\tau^{\mathbf{r}}_{PatId}$ considering the possible deletions of at most one tuple.

3.3 Algorithms for Checking *APE*-FDS

As we proved in Appendix A, given an *APE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\rightarrow} \overline{Z}$ and an instance $\mathbf{r}$ of R, the problem Check-*APE*-FD is NP-Complete in $|\mathbf{r}|$. Then, in principle, there is no asymptotically better algorithm than exploring the whole set of possible subsets $\mathbf{r}'$ of $\mathbf{r}$ with $\frac{|\mathbf{r}|-|\mathbf{r}'|}{|\mathbf{r}|} \leq \epsilon$.

In the following, we provide two algorithms that make use of heuristics, for pruning the search space in order to achieve the tractability for many cases.

The first algorithm is the more general one and it may be applied without assumptions on the input instance $\mathbf{r}$. Such algorithm makes use of two optimization techniques. The first one consists of trying, whenever it is possible, to split the current subset of $\mathbf{r}$ in two subsets, on which the problem may be solved independently (i.e., choices in one subset do not affect those in the other one and vice-versa). The latter optimization technique consists of checking if the current partial solution may not lead to an optimal solution (i.e., a solution $\mathbf{r}'$ where $|\mathbf{r}'|$ is the maximum possible number of tuples that may be kept). If this happens, the subtree is pruned immediately (i.e., we are looking only for optimal solutions).

The second algorithm is applicable under the assumption that we have a bounded and relatively small number of tuples that share the same values for both VT and J which is often the case in clinical domains, as we will discuss later on. In this setting we show how to provide an upper bound value for all the candidate solutions that contain the current partial solution and thus we can apply a pure branch-and-bound approach in order to speed up the algorithm even more.

Before discussing in detail the algorithms and their properties, we need to introduce some basic concepts and features for the representation of tuples through graph-based structures.

3.3.1 Graph-based Structures for Tuple Representation

To this regard we use a suitable graph representation of tuples. A directed graph is a pair $G = (V, E)$, where V is a finite set of nodes and $E \subseteq \binom{V}{2}$ is its edge set. Our graphs are simple, i.e., there are no loops and no parallel edges. Let $U \subseteq V$: we denote by $G|_U = (U, E|_U)$ the subgraph of G induced by U, that is, the graph on node set U such that, for every $u, v \in U$, (u, v) is an edge in $G|_U$ if and only if (u, v) is an edge in G. A *Layered Directed Acyclic Graph* (*L-DAG* for short) is a triple $\mathcal{L}_G = (V, E, l)$ where (V, E) is a DAG and l is a function $l : V \rightarrow \{1, \ldots, p\}$ for some $p \in \mathbb{N}$ where for every $(u, v) \in E$ we have $l(u) < l(v)$. We define the *jump-value* of an edge $(u, v) \in E$ as $jump(u, v) = l(v) - l(u)$. For each $i \in \{1, \ldots, p\}$, we denote with L_i the set $L_i = \{v \in V : l(v) = i\}$. Obviously the notion of induced subgraph naturally extends to *L-DAG*. Given an *L-DAG* $\mathcal{L}_G = (V, E, l)$ and a subset U we define the *L-DAG* induced by U as $\mathcal{L}_G|_U = (U, E|_U, l|_U)$ where $(U, E|_U)$ is the U-induced subgraph of the graph (V, E) and $l|_U : U \rightarrow \mathbb{N}$ with $l|_U(v) = l(v)$ for every $v \in U$. An example of *L-DAG* $\mathcal{L}_G = (V, E, l)$ is given in Figure 3-5. Edges $(v_2, v_7), (v_2, v_9), (v_3, v_7)$ and (v_3, v_9) have jump value equal to 3. For instance, for $\mathcal{L}_G$ we may change the layering function l into l' where $L_i = L'_i$ for every $i = 1, \ldots 4$, $L'_5 = L_5 \setminus \{v_7\}$ and $L'_6 = L_6 \cup \{v_7\}$.

We extend the notion of *L-DAG* with weights, denoting it as *wL-DAG*. A *wL-DAG* is expressed through the tuple $\mathcal{WL}_G = (V, E, l, \mathcal{W})$, where $\mathcal{L}_G = (V, E, l)$ is an *L-DAG* and $\mathcal{W}$

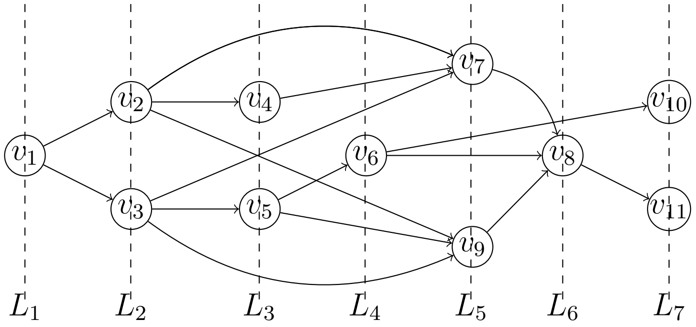


Figure 3-5: An example of *L-DAG*.

is a function $\mathcal{W} : V \rightarrow \mathbb{N}^+$. Let us notice that in our notion of weighted *L-DAG* weights are associated to nodes. Let us now introduce a general problem on *L-DAG*, called *k-Thick Path* (*k-TP* for short).

Problem 1. (*k-TP*). Given an *L-DAG* $\mathcal{L}_G = (V, E, l)$ and a natural number k, determine whether or not there exists a node subset $V' \subseteq V$, such that $|V'| \geq k$ and for every $u, v \in V'$, with $l(u) < l(v)$, there exists a directed path from u to v in $\mathcal{L}_G|_{V'}$.

For instance, if we consider the *L-DAG* in Figure 3-5 we have that the set $V' = \{v_1, v_2, v_3, v_7, v_9, v_8, v_{11}\}$ is a possible solution for k-TP with $k \leq 7$ while $V'' = \{v_1, v_2, v_3, v_4, v_7, v_8, v_{11}\}$ is not a candidate solution since there is no path from v_3 to v_4 and $l(v_3) < l(v_4)$. In a solution we may choose to take more than one node per layer as well as completely ignore all the nodes in a layer. Then, we may see a candidate solution V' as the result of a two-step non-deterministic guess:

1. first we select a set of $p' \leq p$ layers $\{l_1, \ldots l_{p'}\} \subseteq \{1, \ldots, p\}$ (let us assume $l_i < l_j$ for every $1 \leq i < j \leq p'$) which will be all and only the layers which contain at least one node in our solution;

2. for $1 \leq i \leq p'$ we select a non-empty set $V_i \subseteq V$ such that $l(v) = l_i$ for every $v \in V_i$ and for every $(v', v) \in V_{i-1} \times V_i$ we have $(v', v) \in E$.

Going back to the example in Figure 3-5, in V'' condition 2 is violated because by choosing v_4 we choose layer 3 as the non-empty layer following layer 2 but $(v_3, v_4) \notin E$. As a matter of fact, $V'' \setminus \{v_3\}$ (i.e., we choose only v_2 in the layer 2) turns out to be candidate solution. In V' we ignore layers 3 and 4 by not choosing any node in them. Instead, we choose layer 5 as the non-empty layer following layer 2 and everything works just fine.

The *k-TP* problem may naturally be extended to *wL-DAG* by imposing the set V' to satisfy $\Sigma_{v \in V'} \mathcal{W}(v) \geq k$. In [33] we prove that the *k-TP* problem on *wL-DAG* is NP-hard. Our proof can be naturally extended to prove that the non-weighted version of the problem is NP-hard too.

3.3.2 The First Algorithm

Both algorithms rely on the concept of *color* that we will explain through an example in the following. Given an *APE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ and an instance $\mathbf{r}$ of R, let us suppose that we are solving the problem Check-*APE*-FD on such instance with a simple guess-and-check procedure, which makes use of two, initially empty, subsets $\mathbf{r}^+$ (the tuples to be kept in the solution) and $\mathbf{r}^-$ (the tuples to be deleted in the solution) of $\mathbf{r}$. At each step the procedure guesses a tuple t in $\mathbf{r} \setminus (\mathbf{r}^+ \cup \mathbf{r}^-)$ and decides non-deterministically (*guessing phase*) either to update $\mathbf{r}^+$ to $\mathbf{r}^+ \cup \{t\}$ (i.e., t is kept in the current partial solution) or to update $\mathbf{r}^-$ to $\mathbf{r}^- \cup \{t\}$ (i.e, t is deleted in the current partial solution). When $\mathbf{r} = \mathbf{r}^+ \cup \mathbf{r}^-$ (*checking phase*), the procedure returns YES if $\mathbf{r}^+ \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ and $|\mathbf{r}^-| \leq \epsilon \cdot |\mathbf{r}|$, otherwise it returns NO. Hereinafter, we call *partial solution* a triple $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$, such that $(\mathbf{r}^+ \cup \mathbf{r}^-) \subseteq \mathbf{r}$ and $\mathbf{r}^+ \cap \mathbf{r}^- = \emptyset$. If $\mathbf{r} = \mathbf{r}^+ \cup \mathbf{r}^-$ we simply say that $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$ is a *solution*. A solution $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$ is *consistent* if and only if $\mathbf{r}^+ \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$. Given two partial solutions $(\mathbf{r}, \mathbf{r}_1^+, \mathbf{r}_1^-)$ and $(\mathbf{r}, \mathbf{r}_2^+, \mathbf{r}_2^-)$, we say that $(\mathbf{r}, \mathbf{r}_2^+, \mathbf{r}_2^-)$ *extends* $(\mathbf{r}, \mathbf{r}_1^+, \mathbf{r}_1^-)$ if and only if $\mathbf{r}_1^+ \subseteq \mathbf{r}_2^+$ and $\mathbf{r}_1^- \subseteq \mathbf{r}_2^-$.

Is there a way to check whether we are generating an inconsistent solution, possibly without guessing all tuples in $\mathbf{r}$? Violations of the latter constraint (i.e., $|\mathbf{r}^-| \leq \epsilon \cdot |\mathbf{r}|$) are fairly simple to detect during the guessing phase. Indeed, it suffices to check after each insertion in $\mathbf{r}^-$ if $|\mathbf{r}^-|$ exceeds $\epsilon \cdot |\mathbf{r}|$. If it is the case, the procedure may return NO immediately without guessing any further. Violations of the first constraint (i.e., $\mathbf{r}^+ \models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$) during the guessing phase are trickier to detect.

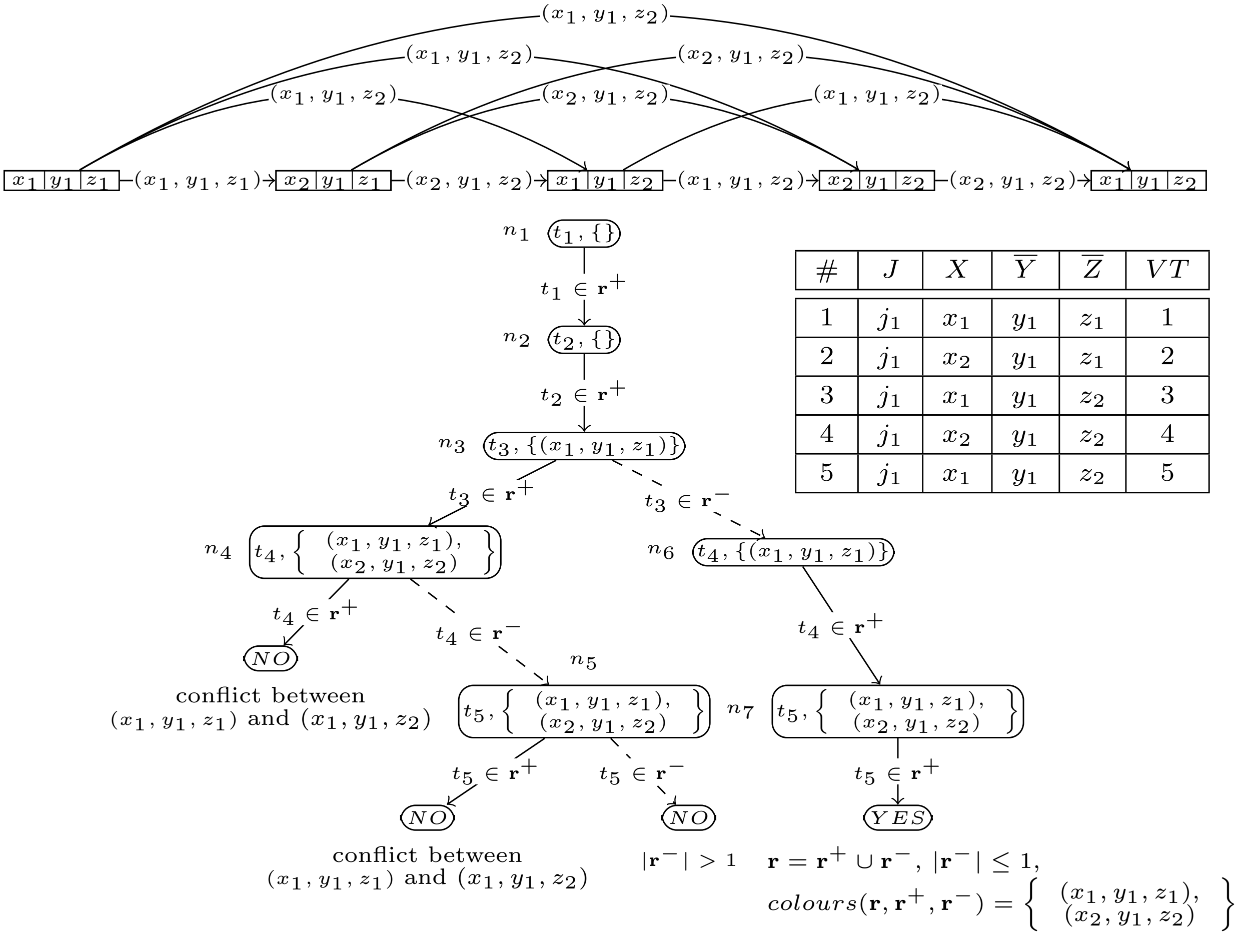
When two tuples t, t' share the same value for attribute J (i.e., $t[J] = t'[J]$), we say that they are in the same J-group and $t[J]$ is the value of the J-group containing t and t'. For the sake of brevity, for a given $j \in Dom(J)$ we will use j-group for denoting the J-group with value j. An *ordered pair*, written *o-pair*, is a pair $(t, t') \in \mathbf{r} \times \mathbf{r}$ such that t and t' are in the same J-group and $t[VT] < t'[VT]$. We say that a pair (t, t') is an *edge* if and only if $0 < t'[VT] - t[VT] \leq k$. Given a triple $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$, an o-pair (t, t') is *active* if and only if $t, t' \in \mathbf{r}^+$ and for every tuple $\hat{t}$ in the same J-group of t if $t[VT] < \hat{t}[VT] < t'[VT]$ we have $\hat{t} \in \mathbf{r}^-$ (i.e., t and t' are selected in the current partial solution and all the tuples between t and t' in the same J-group have been deleted). Given two valid times $vt, vt' \in Dom(VT)$ and a value $j \in Dom(J)$, vt and vt' are consecutive in the j-group if and only if there exists an active o-pair (t, t') with $t[VT] = vt$, $t'[VT] = vt'$, and $t[J] = j$. It is important to observe that we may have two distinct values $j, j' \in Dom(J)$ and two distinct valid times $vt, vt' \in Dom(VT)$ which are consecutive in j-group and not consecutive in j'-group. Moreover, we may have edges (t, t') that are not active and active pairs (t, t') that are not edges, such is the case of active pairs (t, t') with $t'[VT] - t[VT] > k$. A *color* is a tuple c on the schema $C = XYZ$ and we say that two colors (x, y, z) and (x', y', z') are *conflicting* if and only if $x = x'$, $y = y'$, and $z \neq z'$. Given an o-pair (t, t'), its color, denoted by $c(t, t')$, is the tuple $c(t, t') = (t[X], t'[Y], t'[Z])$. Two o-pairs $(t, t'), (t'', t''')$ are *conflicting* if and only if $c(t, t')$ and $c(t'', t''')$ are conflicting.

Theorem 1. Given an *APE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$, an instance $\mathbf{r}$ of R, and a partial solution $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$, if there exist two active and conflicting edges (t, t') and (t'', t'''), then for every solution $(\mathbf{r}, \mathbf{r}_f^+, \mathbf{r}_f^-)$ that extends $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$ it holds $\mathbf{r}_f^+ \not\models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$ (i.e., the solution is inconsistent).

The above theorem guarantees that from a partial solution $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$ that features at least two conflicting edges we cannot reach a consistent solution $(\mathbf{r}, \mathbf{r}_f^+, \mathbf{r}_f^-)$. In such a case we may return immediately NO without considering any further $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$. The colors of a partial solution $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$ are represented by the set $colors(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-) = \{(t[X], t'[Y], t'[Z]) : (t, t')$ is an active edge in $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)\}$. Clearly, the hypothesis of Theorem 1 applies if and only if set $colors(\mathbf{r},\, \mathbf{r}^+, \mathbf{r}^-)$ contains at least two conflicting colors.

Then, by means of colors, our above guess-and-check procedure may be improved by adding the control on the size of $\mathbf{r}^-$ and by keeping updated the current set of colors $colors(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$. Once an insertion of a tuple in either $\mathbf{r}^+$ or $\mathbf{r}^-$ introduces a color c that is conflicting with at least one color in $(\mathbf{r}, \mathbf{r}^+, \mathbf{r}^-)$, the procedure answers NO immediately. An example of how the procedure works is given in Figure 3-6, where we have an instance of 5 tuples with $\epsilon = 0.2$ (i.e., we may delete at most one tuple) and $k = 6$ (all the tuples are in the same window). The execution depicted in Figure 3-6 guesses the values of tuples from the oldest (t_1) to the newest one (t_2) according to the value of VT. First it tries to put the current tuple t in $\mathbf{r}^+$; if no violation arises, it continues; if some violation arises, it tries to insert tuple t in $\mathbf{r}^-$; if no violation arises, it continues, otherwise it goes back to the previous choice (i.e., backtracking). Every internal node is labelled with the current tuple, which will be guessed next; every leaf is labeled either with YES (i.e., the current branch is a solution) or NO (i.e, a violation has arisen); the current set of colors is reported within the node. Nodes are numbered according to their order of appearance. We have that the root is n_1 followed by the introduction of nodes $n_2 \dots n_4$ in this precise order. If we introduce t_4 in the partial solution associated to n_4, we violate the first constraint. Since in n_4 adding t_4 in $\mathbf{r}^-$ does not generate any violation, node n_5 is created as child of n_4. However, node n_5 cannot be extended without introducing a violation in the above constraints. Indeed, if we put t_5 in $\mathbf{r}^+$, we introduce a conflicting color; if we put t_5 in $\mathbf{r}^-$, we exceed the maximum number of allowed deletions. We backtrack to n_4. As all the possible choices have been explored, we backtrack to n_3, where the choice of adding t_3 to $\mathbf{r}^-$ is attempted, generating node n_6. From n_6 we put t_5 in $\mathbf{r}^+$ without violating any constraint and thus we have that $\{t_1, t_2, t_4, t_5\} \models [\Delta_6(\tau_J^R)]X\overline{Y} \overset{0.2}{\to} \overline{Z}$.

Let us now consider in some more detail the first algorithm. Basically, the algorithm works similarly to the previous procedure, except for some trivial technicalities. Two more heuristics have been introduced, to possibly stop earlier, during the exploration of a branch in the tree of computation. The main procedure of the algorithm is reported in Figure 3-9, while auxiliary procedures are reported in Figures 3-7 and 3-8. The algorithm is implemented by function $TupleWiseMin$ that takes 4 arguments. The first argument is $G_\mathbf{r}$, which is derived from $\mathbf{r}$ considering the APE-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ that has to be checked. More precisely, $G_\mathbf{r}$ is an instance of schema $J, X, Y, Z, VT, count$, with $Dom(count) = \mathbb{N}$. We have that $t \in G_\mathbf{r}$ if and only if there exists $t' \in \mathbf{r}$ for which $(t'[J], t'[X], t'[Y], t'[Z]) = (t[J], t[X], t[Y], t[Z])$ and $t[count] = |\{t' \in \mathbf{r} : (t'[J], t'[X], t'[Y], t'[Z]) = (t[J], t[X], t[Y], t[Z])\}|$, that is, we count how many tuples in $\mathbf{r}$ share the same values for attributes J, X, Y, and Z, respectively. The input parameter k is the length for the grouping sliding window. Sets $G_\mathbf{r}^+$ and $G_\mathbf{r}^-$, originally



#	J	X	$\overline{Y}$	$\overline{Z}$	VT
1	j_1	x_1	y_1	z_1	1
2	j_1	x_2	y_1	z_1	2
3	j_1	x_1	y_1	z_2	3
4	j_1	x_2	y_1	z_2	4
5	j_1	x_1	y_1	z_2	5

Figure 3-6: An example of how the use of colors improves a guess and check procedure for solving the problem Check-*APE*-FD.

initialized to $\emptyset$, represent the tuples of $G_{\mathbf{r}}$ that are either kept or deleted in the current solution, respectively. On instances s of schema $J, X, Y, Z, VT, count$, we denote with $||r||$ the sum on the *count* attribute for the tuples in s (i.e., $||s|| = \sum_{t \in s} t[count]$). Finally, $\mathcal{C}$ is a set of *colors* which is initially set to $\emptyset$. A color c is a tuple on the schema X, Y, Z. As we will see, $\mathcal{C}$ keeps track via colors of the constraints introduced so far in the construction of the solution.

Procedure *TupleWiseMin* returns the minimum number of tuples that has to be deleted from $\mathbf{r}$ in order to obtain an instance $\mathbf{r}'$ such that $\mathbf{r}' \models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$. Then if such minimum is less or equal than $\epsilon \cdot |\mathbf{r}|$ we can conclude $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$, else we have $\mathbf{r} \not\models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$. Given $G_{\mathbf{r}}$, $G_{\mathbf{r}}^+$, $G_{\mathbf{r}}^-$, and a set of colors $\mathcal{C}$ we say that an edge $(t, t') \in G_{\mathbf{r}} \times G_{\mathbf{r}}$ is *pending* if and only if the following conditions hold:

1. $t, t' \notin G_{\mathbf{r}}^-$ and $(t[X], t'[Y], z) \notin \mathcal{C}$ for every $z \in Dom(Z)$;

```
procedure InBetween($G_{\mathbf{r}}, t_1, t_2$)
  comment: returns the subset of $G_{\mathbf{r}}$ consisting of all the tuples in the same $J$-group
           of $t_1$ and $t_2$ that feature valid times between $t_1[VT]$ and $t_2[VT]$.
  return $\{t \in G_{\mathbf{r}} : t_1[VT] < t[VT] < t_2[VT] \wedge t[J] = t_1[J]\}$

procedure EdgeConflict?($(t_1, t_2), (t'_1, t'_2)$)
  comment: returns true if and only if the two input edges feature conflicting colors.
  if $t_1[X] = t'_1[X] \wedge t_2[\overline{Y}] = t'_2[\overline{Y}] \wedge t_2[\overline{Z}] \neq t'_2[\overline{Z}]$
    then return true
    else return false

procedure ColorConflict?($(t_1, t_2), \mathcal{C}$)
  comment: returns true if and only if the color of the edge $(t_1, t_2)$ is conflicting with
           at least one color in the set of colors $\mathcal{C}$.
  if $\exists c(c \in \mathcal{C} \wedge t_1[X] = c[X] \wedge t_2[\overline{Y}] = c[\overline{Y}] \wedge t_2[\overline{Z}] \neq c[\overline{Z}])$
    then return true
    else return false

procedure E!($G_{\mathbf{r}}, k, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}$)
  comment: returns the set of all and only active edges in the current partial solution.
  return $\left\{ (t_1, t_2) : \begin{array}{l} t_1[J] = t_2[J] \wedge 0 < t_2[VT] - t_1[VT] \leq k \wedge \\ \{t_1, t_2\} \subseteq G^+_{\mathbf{r}} \wedge InBetween(G_{\mathbf{r}}, t_1, t_2) \subseteq G^-_{\mathbf{r}}) \end{array} \right\}$

procedure E?($G_{\mathbf{r}}, k, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}, \mathcal{C}$)
  comment: returns the set of all and only pending edges in the current partial solution.
  return $\left\{ (t_1, t_2) : \begin{array}{l} t_1[J] = t_2[J] \wedge 0 < t_2[VT] - t_1[VT] \leq k \wedge \{t_1, t_2\} \cap G^-_{\mathbf{r}} = \emptyset \wedge \\ \neg ColorConflict?((t_1, t_2), \mathcal{C}) \wedge InBetween(G_{\mathbf{r}}, t_1, t_2) \cap G^+_{\mathbf{r}} = \emptyset \wedge \\ (\{t_1, t_2\} \cup InBetween(G_{\mathbf{r}}, t_1, t_2)) \nsubseteq (G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}}) \end{array} \right\}$

procedure GroupIndependent?($j, G_{\mathbf{r}}, k, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}, \mathcal{C}$)
  comment: returns true if and only if in the current partial solution pending edges
           involving tuples belonging to the $J$-group with value $j$ are not conflicting
           with the pending edges introduced by other $J$-groups.
  $E_j \leftarrow \{(t_1, t_2) \in E?(G_{\mathbf{r}}, k, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}, \mathcal{C}) : t_1[J] = j\}$
  $\overline{E}_j \leftarrow E?(G_{\mathbf{r}}, k, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}, \mathcal{C}) \setminus E_j$
  if $\exists t_1 \exists t_2 \exists \overline{t}_1 \exists \overline{t}_2 ((t_1, t_2) \in E_j \wedge (\overline{t}, \overline{t}_2) \in \overline{E}_j \wedge EdgeConflict?((t_1, t_2), (\overline{t}_1, \overline{t}_2)))$
    then return false
    else return true
```

Figure 3-7: Auxiliary procedures used by procedures presented in Figures 3-8, 3-9 and 3-11.

```
procedure MaxConsistentSubset($E$)
  comment: returns the maximal subset $E'$ of $E$ such that for every edge $(t'_1, t'_2) \in E'$
           and for every edge $(t_1, t_2) \in E$ we have that $c(t'_1, t'_2)$ and $c(t_1, t_2)$ are not
           conflicting.
  $E' \leftarrow \emptyset$
  for each $(t_1, t_2) \in E$ do
    if $\forall \bar{t}_1 \forall \bar{t}_2 ((\bar{t}_1, \bar{t}_2) \in E \rightarrow \neg EdgeConflict?((t_1, t_2), (\bar{t}_1, \bar{t}_2)))$
      then $E' \leftarrow E' \cup \{(t_1, t_2)\}$
  return $E'$
```

```
procedure Reach?($t_1, t_2, Nodes, Edges$)
  comment: returns true if and only if there exists a path from $t_1$ to $t_2$ in the graph
           $(Nodes, Edges)$. It is a function that checks whether there exists a path
           between two nodes in a graph or not .
  if $\exists(\{\bar{t}_1 \dots \bar{t}_m\} \subseteq Nodes)(\bar{t}_1 = t_1 \wedge \bar{t}_m = t_2 \wedge \forall(1 \leq i < m)((\bar{t}_i, \bar{t}_{i+1}) \in Edges))$
    then return true
    else return false
```

```
procedure ReplacePath?($t, \bar{t}, G_{\mathbf{r}}, k, G^{+}_{\mathbf{r}}, G^{-}_{\mathbf{r}}, \mathcal{C}$)
  comment: returns true if and only if in the current partial solution the consecutive
           valid times $t[VT]$ and $\bar{t}[VT]$ in $t[J]$-group may be safely replaced.
  if $\neg\exists t_1(t_1 \in G_{\mathbf{r}} \wedge t_1[J] = t[J] \wedge t[VT] < t_1[VT] < \bar{t}[VT])$
    then return false
  $N_s \leftarrow \{t_1 \in G^{+}_{\mathbf{r}} : t_1[J] = t[J] \wedge t_1[VT] = t[VT]\}$
  $N_e \leftarrow \{t_1 \in G^{+}_{\mathbf{r}} : t_1[J] = \bar{t}[J] \wedge t_1[VT] = \bar{t}[VT]\}$
  $N_m \leftarrow \{t_1 \in G^{-}_{\mathbf{r}} : t_1[J] = t[J] \wedge t[VT] < t_1[VT] < \bar{t}[VT]\}$
  $N \leftarrow N_s \cup N_e \cup N_m$
  $\tilde{E} \leftarrow \{(t_1, t_2) : t_1 \in N \wedge t_2 \in N \wedge t_1[VT] < t_2[VT] \wedge (t_1 \notin N_s \vee t_2 \notin N_e)\}$
  $Pairs \leftarrow \{(t_1, t_2) : t_1 \in N \wedge t_2 \in N \wedge t_1[VT] + k < t_2[VT] \wedge (t_1 \notin N_s \vee t_2 \notin N_e)\}$
  $\tilde{E}_{ok} \leftarrow (MaxConsistentSubset(E?(G_{\mathbf{r}}, k, G^{+}_{\mathbf{r}}, G^{-}_{\mathbf{r}}, \mathcal{C})) \cap \tilde{E}) \cup Pairs$
  $NotSafe \leftarrow \emptyset$
  for each $(t_1, t_2) \in \tilde{E} \setminus \tilde{E}_{ok}$
    do
      for each $(\bar{t}_1, \bar{t}_2) \in (E?(G_{\mathbf{r}}, k, G^{+}_{\mathbf{r}}, G^{-}_{\mathbf{r}}, \mathcal{C}) \cup E!(G_{\mathbf{r}}, k, G^{+}_{\mathbf{r}}, G^{-}_{\mathbf{r}}) \cup \tilde{E})$
        do
          if $EdgeConflict?((t_1, t_2), (\bar{t}_1, \bar{t}_2))$
            then $NotSafe \leftarrow NotSafe \cup \{(t_1, t_2)\}$
  $\tilde{E} \leftarrow \tilde{E} \setminus NotSafe$
  if $\exists t_1 \exists t_2 \forall \bar{t}_1 \forall \bar{t}_2 (t_1, t_2 \in N_m \wedge \bar{t}_1 \in N_s \wedge \bar{t}_2 \in N_e \wedge (\bar{t}_1, t_1), (t_2, \bar{t}_2) \in \tilde{E} \wedge Reach?(t_1, t_2, N, \tilde{E}))$
    then return true
  return false
```

Figure 3-8: Auxiliary procedures used by procedure *TupleWiseMin* (Figure 3-9).

```
Algorithm 3.3.1: TupleWiseMin($G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+} = \emptyset, G_{\mathbf{r}}^{-} = \emptyset, \mathcal{C} = \emptyset$)

 procedure MaximalPaths?($G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-}, \mathcal{C}$)
   comment: returns true if and only if in the current partial solution for every $J$-group
            $j$-group every pair of consecutive valid times $vt$ and $vt'$ in $j$-group cannot
            be safely replaced.
   if $\exists t_1 \exists t_2 ((t_1, t_2) \in E!(G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-}) \wedge ReplacePath?(t_1, t_2, k, G_{\mathbf{r}}, G_{\mathbf{r}}^{+} G_{\mathbf{r}}^{-}, \mathcal{C}))$
     then return false
     else return true

 procedure IsConsistent?($G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-}, \mathcal{C}$)
   comment: returns true if and only if the current partial solution features consistent
            colors in $\mathcal{C}$ and for every $J$-group $j$-group every pair of consecutive valid
            times $vt$ and $vt'$ in $j$-group cannot be safely replaced.
   if $\exists t_1 \exists t_2 ((t_1, t_2) \in E!(G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-}) \wedge ColorConflict?((t_1, t_2), \mathcal{C}))$
     then return false
   if $MaximalPaths?(G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-}, \mathcal{C})$
     then return true
     else return false

 main
   comment: returns the minimum value $m = \min ||G_{\mathbf{r}} \setminus G'_{\mathbf{r}}||$
            for $G'_{\mathbf{r}}$ in $\{G''_{\mathbf{r}} \subseteq G_{\mathbf{r}} : G'_{\mathbf{r}} \models [\tau_{G_{\mathbf{r}}}^{XYZcount}] X\overline{Y} \rightarrow \overline{Z}\}$.
   if $G_{\mathbf{r}} = \emptyset$ then return $||G_{\mathbf{r}}^{-}||$
   if $\exists j (j \in Dom(J) \wedge GroupIndependent?(j, G_{\mathbf{r}}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-}, \mathcal{C}))$
     then { $\overline{G}_{\mathbf{r}} \leftarrow \{t \in G_{\mathbf{r}} : t[J] = j\}$
          { return ( $TupleWiseCheck(\overline{G}_{\mathbf{r}}, k, G_{\mathbf{r}}^{+} \cap \overline{G}_{\mathbf{r}}, G_{\mathbf{r}}^{-} \cap \overline{G}_{\mathbf{r}}, \mathcal{C})+$
          {          $TupleWiseCheck(G_{\mathbf{r}} \setminus \overline{G}_{\mathbf{r}}, k, G_{\mathbf{r}}^{+} \setminus \overline{G}_{\mathbf{r}}, G_{\mathbf{r}}^{-} \setminus \overline{G}_{\mathbf{r}}, \mathcal{C})$ )
   let $t \in G_{\mathbf{r}}$
   if $IsConsistent?(G_{\mathbf{r}} \setminus \{t\}, k, G_{\mathbf{r}}^{+} \cup \{t\}, G_{\mathbf{r}}^{-}, \mathcal{C})$
     then { $\mathcal{C}' \leftarrow \mathcal{C} \cup \{(t'[X], t''[Y], t''[Z]) : (t', t'') \in E!(G_{\mathbf{r}} \setminus \{t\}, k, G_{\mathbf{r}}^{+} \cup \{t\}, G_{\mathbf{r}}^{-})\}$
          { $m_t \leftarrow TupleWiseCheck(G_{\mathbf{r}} \setminus \{t\}, k, G_{\mathbf{r}}^{+} \cup \{t\}, G_{\mathbf{r}}^{-}, \mathcal{C}')$
     else $m_t \leftarrow +\infty$
   if $IsConsistent?(G_{\mathbf{r}} \setminus \{t\}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-} \cup \{t\}, \mathcal{C})$
     then { $\mathcal{C}' \leftarrow \mathcal{C} \cup \{(t'[X], t''[Y], t''[Z]) : (t', t'') \in E!(G_{\mathbf{r}} \setminus \{t\}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-} \cup \{t\})\}$
          { $m_{\not t} \leftarrow TupleWiseCheck(G_{\mathbf{r}} \setminus \{t\}, k, G_{\mathbf{r}}^{+}, G_{\mathbf{r}}^{-} \cup \{t\}, \mathcal{C}')$
     else $m_{\not t} \leftarrow +\infty$
   return $\min(m_t, m_{\not t})$
```

Figure 3-9: The main procedure for a tuple-wise check of *APE*-FDs. Notice that we use a compact notation for the recursive procedure which is initially called TupleWiseMin($G_{\mathbf{r}}, k$). Here, when $G_{\mathbf{r}}^{+}$, $G_{\mathbf{r}}^{-}$, and $\mathcal{C}$ are omitted in the procedure call, they get their respective default values specified in the procedure declaration (i.e., $\emptyset$ for each of them in this case).

2. for every t'' with $t''[J] = t[J]$ and $t[VT] < t''[VT] < t'[VT]$ we have $t'' \notin G^+_{\mathbf{r}}$;

3. there exists $t'' \in (G_{\mathbf{r}} \cup \{t, t'\}) \setminus (G^-_{\mathbf{r}} \cup G^+_{\mathbf{r}})$ with $t''[J] = t[J]$ and $t[VT] \leq t''[VT] \leq t'[VT]$.

Informally speaking, a pending edge is an edge that is not active in the current partial solution but it may become active during the computation and, if it happens, it introduces a new color in $\mathcal{C}$. In our algorithm, pending edges for the current partial solution are retrieved by procedure $E?$, while active edges are retrieved by procedure $E!$.

Procedure *TupleWiseMin* (Figure 3-9) works as follows. If $G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}} = G_{\mathbf{r}}$, it means that we have obtained a solution without violating any constraint and thus we can return $||G^-_{\mathbf{r}}||$ (i.e., the number of deleted tuples). If $G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}} \neq G_{\mathbf{r}}$, the algorithm guesses a tuple $t \in G_{\mathbf{r}} \setminus (G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}})$ and proceeds as follows. First, it checks if inserting t into $G^+_{\mathbf{r}}$ does not cause any violation of constraints. If so, it stores in m_t the value of the recursive call to *TupleWiseMin* where t belongs to $G^+_{\mathbf{r}}$ and $\mathcal{C}$ has been updated accordingly. By inserting a tuple t in $G^+_{\mathbf{r}}$, the algorithm is asserting that t belongs to the current partial solution, while by inserting t in $G^-_{\mathbf{r}}$ the algorithm is asserting that t does not belong to the current partial solution. If a constraint is violated, the algorithm stores in m_t the value $+\infty$, which means that t may not be kept in the current solution.

Then, it checks if inserting t into $G^-_{\mathbf{r}}$ does not cause any violation in the constraints. If it is the case, it stores in $m_{\not t}$ the value of the recursive call to *TupleWiseMin*, where $G^-_{\mathbf{r}}$ and $\mathcal{C}$ are updated accordingly. If a constraint is violated, the algorithm stores in $m_{\not t}$ the value $+\infty$, which means that t must be kept in the current partial solution. In procedure *TupleWiseMin* the only way in which a constraint may be violated is that, after the insertion a tuple t in $G^+_{\mathbf{r}}$ (resp. $G^-_{\mathbf{r}}$), an edge (t', t'') turns out to be active and its color $(t'[X], t''[Y], t''[Z])$ turns out to be conflicting with at least one color in $\mathcal{C}$.

As pointed out by the example in Figure 3-6, checking each step for consistency is itself an optimization, even if it is trivial, since it allows us to prune entire subtrees in the tree of computations without exploring them. We propose here two further optimizations for this procedure. The first one allows us to restrict the search space by splitting the problem into independent sub-problems in a divide-and-conquer fashion. Let us suppose that at a certain step of our computation there exists a value $j \in \{t[J] : t \in G_{\mathbf{r}}\}$, for which for each pair of conflicting pending edges (t, t') and $(\hat{t}, \hat{t}')$ we have that either all $t, t', \hat{t}$, and $\hat{t}'$ belong to the j-group or all $t, t', \hat{t}$, and $\hat{t}'$ do not belong to the j-group (such condition is verified by sub-procedure *GroupIndependent?* reported in Figure 3-8.) Let $\overline{G}_{\mathbf{r}} = \{t : t[J] = j\}$. As every edge involving tuples in the j-group is not conflicting with every edge that may be introduced outside the j-group, then we can split the problem into the two sub-problems $(\overline{G}_{\mathbf{r}}, k, G^+_{\mathbf{r}} \cap \overline{G}_{\mathbf{r}}, G^-_{\mathbf{r}} \cap \overline{G}_{\mathbf{r}})$ and $(G_{\mathbf{r}} \setminus \overline{G}_{\mathbf{r}}, k, G^+_{\mathbf{r}} \setminus \overline{G}_{\mathbf{r}}, G^-_{\mathbf{r}} \setminus \overline{G}_{\mathbf{r}})$. Such problems are independent and may be solved separately. The resulting value for the solution is the sum of the values returned by *TupleWiseMin* applied to both the two sub-problems. Let $H = |G_{\mathbf{r}} \setminus (G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}})|$ and $h = |\{t \in (G_{\mathbf{r}} \setminus (G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}})) : t[J] = j\}|$. In this case, the upper bound of the complexity at the current step of computation drops from $\mathcal{O}(2^H)$ to $\mathcal{O}(2^{H-h} + 2^h)$.

The second optimization allows us to prune a sub-tree of computation even before a contradiction arises. It verifies, in many cases, whether every possible solution that may be built starting from the current partial one turns to be not minimal. Suppose that there exists an active o-pair (t, t') in a partial solution $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$, such that there exists $\hat{t} \in G_{\mathbf{r}}$ in the same J-group of t with $t[VT] < \hat{t}[VT] < t'[VT]$. By definition of active o-pair, we have that $\hat{t}$ belongs to $G^-_{\mathbf{r}}$ as well as every tuple $\hat{t}'$ in the same J-group of t with $t[VT] < \hat{t}'[VT] < t'[VT]$. Here the additional condition is that there exists at least one of such tuples. Given a partial solution $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$, we define set $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}) = \{(x, y, z)$: there exists an active edge (t, t') with $c(t,\ t') = (x, y, z)\}$. Let us define the set of colors $pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}) = \{(x, y, z)$: there exists a pending edge (t, t') with $c(t, t') = (x, y, z)\}$, which collects all and only the colors that may be introduced later on in the current computation.

A color (x, y, z) is *safe* in (vt, vt', j) if and only if one of the following three conditions hold:

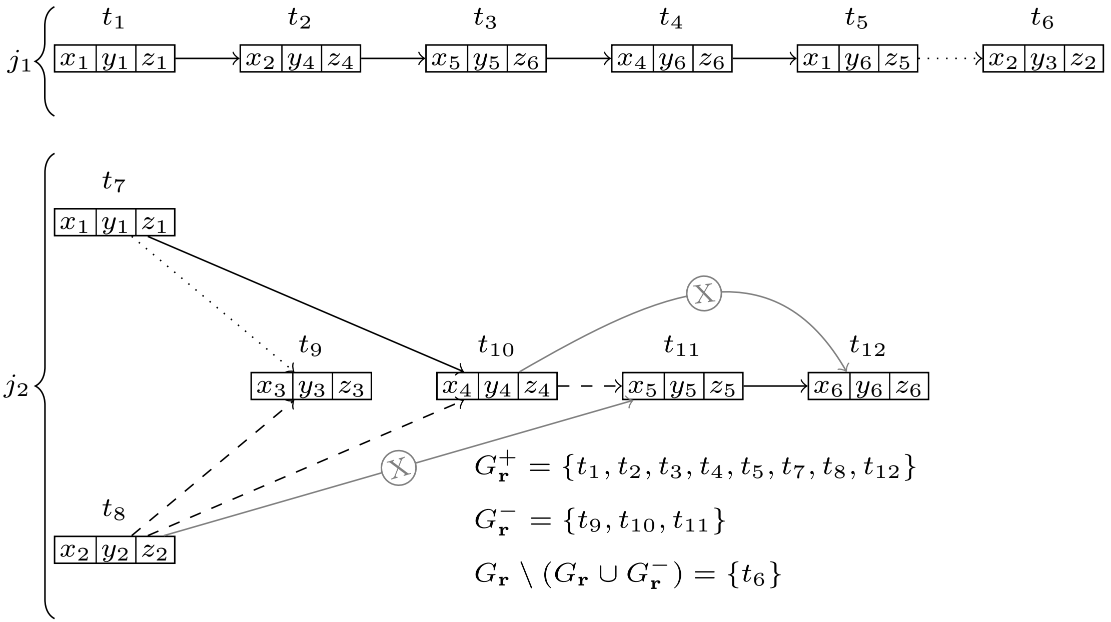
1. $(x, y, z) \in colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$;

2. every color $(x, y, z') \in pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ satisfies $z' = z$ (i.e., (x, y, z) is a pending color and there is no pending color that is conflicting with (x, y, z));

3. the color is not conflicting with any color in $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}) \cup pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ and do not exist two tuples $\hat{t}, \hat{t}' \in (G^+_{\mathbf{r}} \cup G^-_{\mathbf{r}}) \cap \{\hat{t}'' \in G_{\mathbf{r}} : \hat{t}''[J] = j \wedge vt \leq \hat{t}'' \leq vt'\}$, such that $(\hat{t}, \hat{t}')$ is an edge and the color $(\hat{t}[X], \hat{t}'[Y], \hat{t}'[Z])$ is conflicting with (x, y, z).

The three conditions above imply that if a color is safe in (vt, vt', j) then it is neither in conflict with a color in $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ nor with a color in $pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$. However, this is just a necessary but not sufficient condition. Let us consider the example shown in Figure 3-10 and assume that $k \geq 7$ (i.e., every o-pair in the example is also an edge). We have that the active edges are $(t_1, t_2), (t_2, t_3), (t_3, t_4)$, and (t_4, t_5) for the j_1-group and $(t_7, t_{12}), (t_8, t_{12})$ for the j_2-group, since we have $t_9, t_{10}, t_{11} \in G^-_{\mathbf{r}}$. Thus, we have $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}}) = \{(x_1, y_4, z_4), (x_2, y_5, z_6), (x_5,\ y_6, z_6), (x_4, y_6, z_5),\ (x_1, y_6, z_6), (x_2, y_6, z_6)\}$ and, since we have to decide the status of tuple t_6, we have $pending(G_{\mathbf{r}}, G^+_{\mathbf{r}},\ G^-_{\mathbf{r}}) = \{(x_1, y_3, z_2)\}$. Suppose that we are interested in the colors which are safe in $(1, 5, j_2)$.

For instance $c(t_7, t_9) = (x_1, y_3, z_3)$ (i.e., the dotted edge in the j_2-group in Figure 3-10) is not safe in $(1, 5, j_2)$ because it does not belong neither to $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ nor to $pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ (conditions 1 and 2) and it is in conflict with the unique color $(x_1, y_3, z_2) \in pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$. On the other hand, colors $c(t_8, t_9) = (x_2, y_3, z_3)$, $c(t_8, t_{10}) = (x_2, y_4, z_4)$ and $c(t_{10}, t_{11}) = (x_4, y_5, z_5)$ (i.e., the dashed edges in Figure 3-10) are safe in $(1, 5, j_2)$, because they are neither in conflict with colors either in $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ or in $pending(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ and there are no other edges in j_2 with valid times between 1 and 5 that exhibit either (x_2, y_3), or (x_4, y_5), or (x_2, y_3) as first two components of their colors (thus condition 3 applies to these colors). Colours $c(t_7, t_{10}) = (x_1, y_4, z_4)$ and $c(t_{11}, t_{12}) = (x_5, y_6, z_6)$ (i.e., the continuous edges in j_2-group in Figure 3-10) are safe in $(1, 5, j_2)$, because they belong to $colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ and thus both satisfy condition 1. Finally, colors $c(t_8, t_{11}) = (x_2, y_5, z_5)$, and $c(t_8, t_{10}) = (x_4, y_6, z_6)$ are not safe in $(1, 5, j_2)$ (i.e., the X-labeled edges in Figure 3-10), because they are in conflict with two colors in


j_1
t_1 x_1 y_1 z_1
t_2 x_2 y_4 z_4
t_3 x_5 y_5 z_6
t_4 x_4 y_6 z_6
t_5 x_1 y_6 z_5
t_6 x_2 y_3 z_2
j_2
t_7 x_1 y_1 z_1
t_9 x_3 y_3 z_3
t_{10} x_4 y_4 z_4
t_{11} x_5 y_5 z_5
t_{12} x_6 y_6 z_6
t_8 x_2 y_2 z_2
X
X
$G^+_{\mathbf{r}} = \{t_1, t_2, t_3, t_4, t_5, t_7, t_8, t_{12}\}$
$G^-_{\mathbf{r}} = \{t_9, t_{10}, t_{11}\}$
$G_{\mathbf{r}} \setminus (G_{\mathbf{r}} \cup G^-_{\mathbf{r}}) = \{t_6\}$

#	J	X	Y	Z	VT	count
1	j_1	x_1	y_1	z_1	2	1
2	j_1	x_2	y_4	z_4	4	1
3	j_1	x_5	y_5	z_6	5	1
4	j_1	x_4	y_6	z_6	6	1
5	j_1	x_1	y_6	z_5	7	1
6	j_1	x_2	y_3	z_2	6	1
7	j_2	x_1	y_1	z_1	1	1
8	j_2	x_2	y_2	z_2	2	1
9	j_2	x_3	y_3	z_2	3	1
10	j_2	x_4	y_4	z_3	4	1
11	j_2	x_5	y_5	z_4	5	1
12	j_2	x_6	y_6	z_5	6	1

Figure 3-10: An example of how a partial solution may be improved.

$colors(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$; more precisely, (x_2, y_5, z_5) is in conflict with (x_2, y_5, z_6) and (x_4, y_6, z_6) is in conflict with (x_4, y_6, z_5).

Given a partial solution $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ and the triple (vt, vt', j), a (vt, vt', j)*-replace DAG* is a DAG (V, E) where $V = \{t \in G^+_{\mathbf{r}} : t[VT] = vt \wedge t[J] = j\} \cup \{t \in G^-_{\mathbf{r}} : vt < t[VT] < vt' \wedge t[J] = j\} \cup \{t \in G^+_{\mathbf{r}} : t[VT] = vt' \wedge t[J] = j\}$ and

$$E = \begin{array}{c} \left\{ (t, t') \in V \times V : \begin{array}{c} (t[VT] \neq vt \vee t'[VT] \neq vt') \wedge t[VT] < t'[VT] \wedge \\ c(t, t') \text{ is safe in } (vt, vt', j) \end{array} \right\} \\ \cup \\ \{(t, t') \in V \times V : (t[VT] \neq vt \vee t'[VT] \neq vt') \wedge t'[VT] - t[VT] > k\} \end{array}$$

A node $t \in V$ is a *starting node* (resp. *ending node*) if and only if $vt < t[VT] < vt'$ and, for every $t' \in V$ with $t'[VT] = vt$ (resp. $t'[VT] = vt'$), we have $(t', t) \in E$ (resp. $(t, t') \in E$). A *replace path* in a (vt, vt', j)-replace DAG (V, E) is any path $t_1 \dots t_m$ in (V, E) for which t_1 is a starting node and t_m is an ending node. We say that vt and vt' in j can be *safely replaced* if and only if there exists a replace path in the (vt, vt', j)-replace DAG (V, E). Figure 3-10 depicts the $(1, 5, j_2)$-replace DAG, where t_{10} is the only initial node that is not an ending one, and t_{11} is the only ending node that is not a starting one. Since t_{10} is connected to t_{11} we have that $t_{10}t_{11}$ is a replace path in the $(1, 5, j_2)$-replace DAG and thus 1 and 5 can be safely replaced in j_2. Using the above definitions of replace DAGs/paths we can provide the following result.

Theorem 2. Given a partial solution $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$, if there exists a group j with two consecutive valid times vt and vt' such that vt and vt' can be safely replaced in j, then every consistent solution that follows $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ is not optimal.

The proof of the theorem is straightforward. Let us suppose that $t_1 \dots t_m$ is a replace path in the (vt, vt', j)-replace DAG. By definition we have $t_1, \dots, t_m \in G^-_{\mathbf{r}}$. It suffices to take any consistent solution $(G_{\mathbf{r}}, \overline{G}^+_{\mathbf{r}}, \overline{G}^-_{\mathbf{r}})$ that follows $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ and such that $(G_{\mathbf{r}}, \overline{G}^+_{\mathbf{r}} \cup \{t_1, \dots, t_m\}, \overline{G}^-_{\mathbf{r}} \setminus \{t_1, \dots, t_m\})$ is still a consistent solution. Non-optimality immediately follows.

We take advantage of Theorem 2 by pruning every computation rooted in a partial solution $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ that features a J-group j-group and two consecutive valid times vt and vt' in the j-group, such that vt and vt' can be safely replaced in j. Verifying whether such condition applies or not may be performed in polynomial time. In procedure *TupleWiseMin*, this optimization is realized by sub-procedures *MaximalPaths?* and *ReplacePath?* reported in Figure 3-8 and 3-9, respectively.

3.3.3 The Second Algorithm

Let us now propose another algorithm, reported in Figure 3-13, for solving problem Check-*APE*-FD. Such an algorithm, whose main procedure is called *EdgeWiseMin*, strongly differs from *TupleWiseMin* in approaching the problem. In principle it works better, but it may work only under a quite reasonable assumption on the input, which we will discuss in detail later on.

At every step, procedure *EdgeWiseMin*, instead of guessing if a tuple belongs to the current partial solution, guesses if a color is forbidden or allowed in the current partial solution. Informally, forbidding a color (x, y, z) means avoiding all the active edges $(t, t') \in G_{\mathbf{r}} \times G_{\mathbf{r}}$ for which $c(t, t') = (x, y, z)$. On the other hand, allowing a color (x, y, z) means forbidding all the active edges $(t, t') \in G_{\mathbf{r}} \times G_{\mathbf{r}}$ whose colors are conflicting with (x, y, z). In order to do that, we introduce the concept of *color-partial solution.* A color-partial solution is a triple $(G_{\mathbf{r}}, \mathcal{C}^+, \mathcal{C}^-)$, such that $\mathcal{C}^+, \mathcal{C}^- \subseteq Dom(X) \times Dom(Y) \times Dom(Z)$ are disjoint subsets of colors (i.e, $\mathcal{C}^+ \cap \mathcal{C}^- = \emptyset$) and for every pair of colors $(x, y, z), (x', y', z') \in \mathcal{C}^+$ (x, y, z) is not conflicting with (x', y', z') (i.e, if $x' = x$ and $y' = y$ then $z' = z$).

Let $CPS = (G_{\mathbf{r}}, \mathcal{C}^+, \mathcal{C}^-)$ be a color-partial solution: we say that a solution $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ is *induced by CPS* if and only if the two following conditions hold:

```
procedure SOURCESINKSHORTESTPATH($Nodes, Edges, Time, Weight$)
  comment: returns the shortest path from a $source$ node to a $sink$ node in a Weighted
           DAG ($Nodes, Edges$). The weight of each edge is provided by the function
           $Weight$. The topological order of the elements of $Nodes$ is provided by the
           function $Time$ whch is used for identifying the $source$ ($Time(source) =$
           $-\infty$) and the $sink$ ($Time(sink) = +\infty$) nodes. Finally, the shortest path is
           returned as a subset of $Nodes$.
  for each $n \in Nodes$
    $Predecessor(n) \leftarrow nil$
    $Value(n) \leftarrow +\infty$
  $Value(source) \leftarrow 0$
  $current\_time \leftarrow \min(Img(Time) \setminus \{-\infty\})$
  while $current\_time < +\infty$
    do
      for each $n\{n' \in Nodes : Time(n') = current\_time\}$
        do
          $Value(n) \leftarrow \min_{(n',n) \in Edges} Value(n') + Weight(n, n')$
          $Predecessor(n) \leftarrow \min_{(n',n) \in Edges} n'$
      $current\_time \leftarrow \min\{time \in Img(Time) : time > current\_time\}$
  $current\_node \leftarrow Predecessor(sink)$
  $shortest\_path \leftarrow \emptyset$
  while $current\_node \neq source$
    do
      $shortest\_path \leftarrow shortest\_path \cup \{current\_node\}$
      $current\_node \leftarrow Predecessor(current\_node)$
  return $shortest\_path$
```

Figure 3-11: Auxiliary procedure for the main ones in Figure 3-13. *SourceSinkShortestPath* returns the shortest path from source to sink on the DAG provided by *BuildDag*. The solution is given as a set of nodes (i.e., subsets of $G_{\mathbf{r}}$) and it omits *source* and *sink* nodes.

```
procedure BuildDag($G_{\mathbf{r}}, k, \mathcal{C}^+, \mathcal{C}^-$)
    comment: given a set $G_{\mathbf{r}}$ for which there exists $j \in Dom(J)$ such that for every $t \in G_{\mathbf{r}}$ we have $t[J] = j$. Let $CPS = (G_{\mathbf{r}}, \mathcal{C}^+, \mathcal{C}^-)$ The procedure returns the unfolded DAG for $\mathcal{L}^j_{CPS}$.
    $Nodes \leftarrow \{S \subseteq G_{\mathbf{r}} : S \neq \emptyset \wedge \forall t \forall t'(t \in S \wedge t' \in S \rightarrow t[VT] = t'[VT])\}$
    $EndNodes \leftarrow \{source, sink\}$
    $AllNodes = Nodes \cup EndNodes$
    let $Time : AllNodes \rightarrow Dom(VT) \cup \{-\infty, +\infty\}$ s.t. $Time(S) = \begin{cases} t[VT] & S \in Nodes \wedge \\ & t \in S \\ -\infty & S = source \\ +\infty & S = sink \end{cases}$
    $Edges \leftarrow \begin{array}{l} \{(source, S) : S \in Nodes\} \cup \{(S, sink) : S \in Nodes\} \\ \cup \{(S, S') : S \in Nodes, S' \in Nodes, Time(S') - Time(S) > k\} \end{array}$
    $Edges \leftarrow \left\{(S, S') : \forall t \forall t' \left(\left(\begin{array}{c} t \in S \wedge t' \in S' \wedge \\ t'[VT] - t[VT] > 0 \end{array}\right) \rightarrow \left(\begin{array}{l} \neg ColorConflict((t[X], t'[Y], t'[Z]), \mathcal{C}^+) \\ \wedge (t[X], t'[Y], t'[Z]) \notin \mathcal{C}^- \end{array}\right)\right)\right\}$
    let $Weight : Edges \rightarrow \mathbb{N}$ s.t.
    $Weight(S, S') = \begin{cases} \sum_{t \in G_{\mathbf{r}} : Time(S) < t[VT] \leq Time(S') \wedge t \notin S'} ||t|| & \text{if } S, S' \in Nodes \\ \sum_{t \in G_{\mathbf{r}} : t[VT] \leq Time(S') \wedge t \notin S'} ||t|| & \text{if } S = source \\ \sum_{t \in G_{\mathbf{r}} : t[VT] > Time(S) \wedge t \notin S'} ||t|| & \text{if } S' = sink \end{cases}$
    return $(Nodes \cup EndNodes, Edges, Time, Weight)$
```

Figure 3-12: Auxiliary procedure for the main ones in Figure 3-13. *BuildDag* build a single-source-single-sink DAG which nodes are non-empty subsets of $G_{\mathbf{r}}$. Each subset is formed by tuples sharing the same value for VT and thus function *Time* is well defined.

Algorithm 3.3.2: $\textsc{EdgeWiseMin}(G_{\mathbf{r}}, k, \mathcal{C}^{+} = \emptyset, \mathcal{C}^{-} = \emptyset, \mathit{optimal} = +\infty)$

```
procedure PartialSolution($G_{\mathbf{r}}, k, \mathcal{C}^{+}, \mathcal{C}^{-}$)
  comment: let $CPS = (G_{\mathbf{r}}, \mathcal{C}^{+}, \mathcal{C}^{-})$, the procedure returns a pair $(Value, Colors)$ for
           which $Value = \mathit{val}(CPS)$ and there exists a minimal solution $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^{-}_{\mathbf{r}},$
           $G^{-}_{\mathbf{r}})$ induced by $(G_{\mathbf{r}}, \mathcal{C}^{+}, \mathcal{C}^{-})$ for which $Colors = \mathit{colors}(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^{-}_{\mathbf{r}}, G^{-}_{\mathbf{r}})$.
  $Value \leftarrow 0$
  $Colors \leftarrow \emptyset$
  for each $j \in \{t[J] : t \in G_{\mathbf{r}}\}$
    do
      $G^{j}_{\mathbf{r}} \leftarrow \{t \in G_{\mathbf{r}} : t[J] = j\}$
      $ShortestPath \leftarrow SourceSinkShortestPath(BuildDAG(G^{j}_{\mathbf{r}}, k, \mathcal{C}^{+}, \mathcal{C}^{-}))$
      $\overline{G}_{\mathbf{r}} \leftarrow \bigcup_{S \in ShortestPath} S$
      $Value \leftarrow Value + \sum_{t \in \overline{G}_{\mathbf{r}}} ||t||$
      $ActiveEdges \leftarrow \left\{ (t, t') : \begin{array}{l} t \in \overline{G}_{\mathbf{r}} \wedge t' \in \overline{G}_{\mathbf{r}} \wedge 0 < t'[VT] - t[VT] \leq k \wedge \\ \forall t''(t'' \in \overline{G}_{\mathbf{r}} \rightarrow t''[VT] \leq t[VT] \vee t''[VT] \geq t'[VT]) \end{array} \right\}$
      $Colors \leftarrow Colors \cup \{(t[X], t'[Y], t'[Z]) : (t, t') \in ActiveEdges\}$
  return $(Value, Colors)$

main
  comment: returns the minimum value $m = \min ||G_{\mathbf{r}} \setminus G'_{\mathbf{r}}||$
           for $G'_{\mathbf{r}}$ in $\{G''_{\mathbf{r}} \subseteq G_{\mathbf{r}} : G'_{\mathbf{r}} \models [\tau^{XYZcount}_{G_{\mathbf{r}}}]X\overline{Y} \rightarrow \overline{Z}\}$.
  $m \leftarrow 0$
  $Colors \leftarrow \emptyset$
  for each $g \in \{g : \exists(t \in G_{\mathbf{r}})(t[J] = g)\}$
    do
      $(Value, C) \leftarrow PartialSolution(Gbr, k, \mathcal{C}^{+}, \mathcal{C}^{-})$
      $m \leftarrow m + Value$
      $Colors \leftarrow Colors \cup C$
  if $m \geq \mathit{optimal}$ then return $\mathit{optimal}$
  for each $c \in Colors$
    for each $c' \in Colors$
      if $c[X] = c'[X] \wedge c[\overline{Y}] = c'[\overline{Y}] \wedge c[\overline{Z}] \neq c'[\overline{Z}]$
        then
          $m_{c} \leftarrow EdgeWiseMin(G_{\mathbf{r}}, k, \mathcal{C}^{+} \cup \{c\}, \mathcal{C}^{-}, \mathit{optimal})$
          $m_{\neg c} \leftarrow EdgeWiseMin(G_{\mathbf{r}}, k, \mathcal{C}^{+}, \mathcal{C}^{-} \cup \{c\}, \mathit{optimal})$
          $m \leftarrow \min(m_{c}, m_{\neg c})$
          return $\min(m, \mathit{optimal})$
  return $m$
```

Figure 3-13: The main procedure for the edge-wise checking of a *APE*-FD $[\Delta_k(\tau^{XYZcount}_{G_{\mathbf{r}}})]X\overline{Y} \overset{\varepsilon}{\rightarrow} \overline{Z}$. The procedure returns the minimum number of tuples to delete in $\mathbf{r}$ in order to obtain an instance $\mathbf{r}' \subseteq \mathbf{r}$ such that $\mathbf{r}' \models [\Delta_k(\tau^{R}_{\mathbf{r}})]X\overline{Y} \rightarrow \overline{Z}$. Like procedure *TupleWiseMin* of Figure 3-9 the initial call to the recursive procedure is $EdgeWiseMin(G_{\mathbf{r}}, k)$ with $\mathcal{C}^{+}$, $\mathcal{C}^{-}$, and *optimal* initialized to their respective default values.

1. for every color (x, y, z) in $\mathcal{C}^+$ and for each edge $(t, t') \in G_{\mathbf{r}} \times G_{\mathbf{r}}$ for which $c(t, t') = (x, y, z')$, if (t, t') is active then $z' = z$.

2. for every color (x, y, z) in $\mathcal{C}^-$ and for each edge $(t, t') \in G_{\mathbf{r}} \times G_{\mathbf{r}}$, if $c(t, t') = (x, y, z)$ then (t, t') is not active in $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$. It means that one of the following two conditions holds:

   - $t \in G^-_{\mathbf{r}}$ or $t' \in G^-_{\mathbf{r}}$;
   - there exists a tuple $t'' \in G^+_{\mathbf{r}}$ such that $t[VT] < t''[VT] < t'[VT]$ and $t''[J] = t[J]$ ($t[VT]$ and $t'[VT]$ are not consecutive in the current partial solution for the J-group with value $t[J]$).

An induced solution $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ by CPS is *minimal* if and only if an induced solution $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus \overline{G}^-_{\mathbf{r}}, \overline{G}^-_{\mathbf{r}})$ by CPS does not exist with $|G^-_{\mathbf{r}}| > |\overline{G}^-_{\mathbf{r}}|$. In this case we say that $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ is an *induced minimal solution.* Since all the induced minimal solutions by CPS have the same size, we say that the *value of* CPS, denoted by $val(CPS)$, is the value $||G^-_{\mathbf{r}}||$, where $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ is a minimal induced partial solution.

In an opposite way from *TupleWiseMin*, in this algorithm color-partial solutions induce complete minimal solutions. However such solutions may be inconsistent. The algorithm tries to obtain consistency by either forcing or forbidding one color at a time. This is done by means of sets $\mathcal{C}^+$ and $\mathcal{C}^-$, which are both initialized to $\emptyset$ at the beginning of the procedure. As we informally said above, if a color (x, y, z) belongs to $\mathcal{C}^+$, it means that the current partial solution must avoid all the active edges (t, t') such that $t[X] = x$, $t'[Y] = y$, and $t'[Z] \neq z$; if a color (x, y, z) belongs to $\mathcal{C}^-$, it means that the current partial solution must avoid all the active edges (t, t') such that $t[X] = x$, $t'[Y] = y$, and $t'[Z] = z$.

As a general overview of the algorithm, let us consider the following simplified procedure (let $CPS = (G_{\mathbf{r}}, \mathcal{C}^+, \mathcal{C}^-)$ be the current color-partial solution):

1. compute a minimal solution $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ induced by CPS;

2. if $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ is consistent, return $val(CPS) = ||G^-_{\mathbf{r}}||$;

3. if $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$ is inconsistent, let (t, t') be an active edge in $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}},\ G^-_{\mathbf{r}})$, such that there exists an active edge (t'', t''') in $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$, which is conflicting with (t, t'). Then, return the minimum value between those returned by two recursive calls, one where $\mathcal{C}^+$ is updated to $\mathcal{C}^+ \cup \{(t[X], t'[Y], t'[Z])\}$, and the other one where $\mathcal{C}^-$ is updated to $\mathcal{C}^- \cup \{(t[X], t'[Y], t'[Z])\}$.

Two observations are omitted in the above procedure w.r.t. function *EdgeWiseMin*. The first one is that the procedure does not take into account the fact that the value $||G^-_{\mathbf{r}}||$ of the color-partial solution computed at point 1. is a lower bound for the optimal solution that may be achieved in the current branch of the computation. Procedure *EdgeWiseMin* uses it in a classical branch-and-bound fashion by propagating the value of the current optimal solution (if any) in the tree of recursive calls (in Figure 3-13 this is done by means of parameter *optimal*). In step 1., if the computed value $||G_{\mathbf{r}}^{-}||$ is greater than the optimal one, we immediately return from the recursive call, because no better solution may be found.

The last omitted observation regards how the value of the color-partial solution $val(CPS)$ is computed, where $CPS = (G_{\mathbf{r}}, \mathcal{C}^{+}, \mathcal{C}^{-})$. For every J-group in $G_{\mathbf{r}}$, i.e. any set of tuples having value j for attribute J, we build the following *wL-DAG* $\mathcal{L}_{CPS}^{j} = (V^j, E^j, \mathcal{L}^j, \mathcal{W}^j)$ where $V^j = \{t \in G_{\mathbf{r}} : t[J] = j\}$. For each $t \in V^j$ we have $\mathcal{W}^j(t) = t[count]$ and $\mathcal{L}^j(t) = t[VT]$, and

$$E^j = \begin{array}{c} \{(t,t') \in V^j \times V^j : t'[VT] - t[VT] > k\} \\ \cup \\ \{(t,t') \in V^j \times V^j : 0 < t'[VT] - t[VT] \leq k \wedge c(t,t') \in \mathcal{C}^{+}\} \\ \cup \\ \left\{ (t,t') \in V^j \times V^j : \begin{array}{l} 0 < t'[VT] - t[VT] \leq k \wedge c(t,t') \notin \mathcal{C}^{-} \wedge \\ {\scriptstyle \forall x \forall y \forall z ((x,y,z) \in \mathcal{C}^{+} \rightarrow c(t,t') \text{ is not conflicting with } (x,y,z)} \end{array} \right\} \end{array}$$

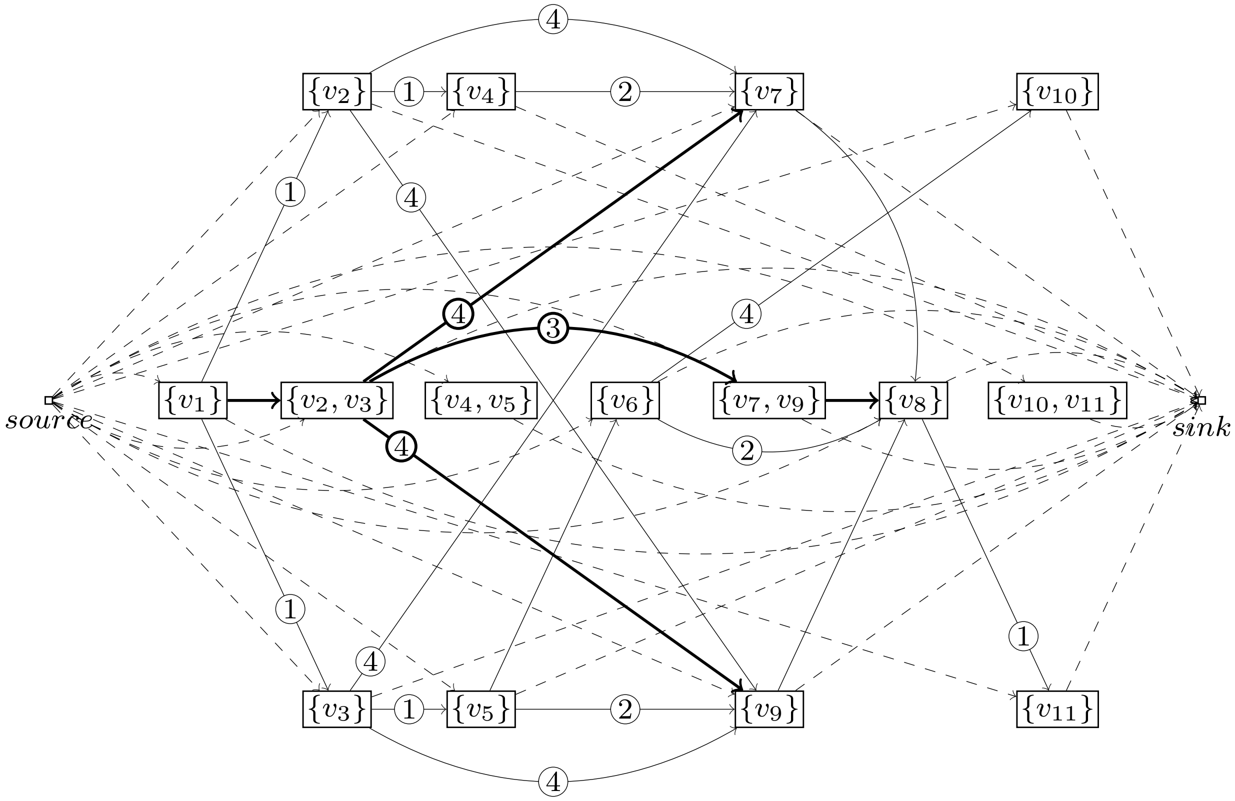
Let $J(G_{\mathbf{r}})$ be the set $\{j : \exists t (t \in G_{\mathbf{r}} \wedge t[J] = j)\} = \{j_1, \ldots, j_h\}$. For every $1 \leq i \leq h$ let M_{CPS}^{i} be the maximum value for which the *wL-DAG* $\mathcal{L}_{CPS}^{j_i}$ admits an M_{CPS}^{i}-thick path. The following result is straightforward.

Theorem 3. For every color-partial solution $CPS = (G_{\mathbf{r}}, \mathcal{C}^{+}, \mathcal{C}^{-})$ we have

$$val(CPS) = ||G_{\mathbf{r}}|| - \sum_{j \quad J(G_{\mathbf{r}})} M_{CPS}^{j}$$

Theorem 3 tells us that for computing $val(CPS)$ we need to compute for every $j \in J(G_{\mathbf{r}})$ the maximum value M_{CPS}^{j}, for which $\mathcal{L}_{CPS}^{j}$ admits an M_{CPS}^{j}-thick path. Given a *wL-DAG* $\mathcal{L}_{\mathcal{G}}$, we will call MAX-ThickPath (Max-TP for short) the problem of finding the maximum M for which $\mathcal{L}_{\mathcal{G}}$ admits a M-Thick Path. Max-TP may be solved by a simple dichotomic search having a decision procedure that solves the problem M-TP (Problem 1), which is NP-Complete [33]. Here our assumption comes into play and allows us to find M_{CPS}^{j} for every $j \in J(G_{\mathbf{r}})$ in a "reasonable" time. Indeed, in the instances that are used to prove the NP-completeness the number of nodes in any layer, roughly corresponding to the number of tuples of the given relation at a corresponding time point, is supposed to increase as the number of time points/layers increases.

This is not the case in many daily applications, especially in the clinical domain, where we may have a great number of tuples but scattered along the time-line. In the following we will provide a formal definition of our assumption. Given $j \in J(G_{\mathbf{r}})$ we define $VT(G_{\mathbf{r}}, j) = \{vt : \exists t (t \in G_{\mathbf{r}} \wedge t[VT] = vt \wedge t[J] = j)\}$, $MaxLevel(G_{\mathbf{r}}, j) = \max_{vt \in VT(G_{\mathbf{r}}, j)} |\{t : t[J] = j \wedge t[VT] = vt\}|$, $MaxCount(G_{\mathbf{r}}, j) = \max_{vt \in VT(G_{\mathbf{r}}, j)} \left(\sum_{t \in G_{\mathbf{r}} \wedge t[VT] = v \wedge t[J] = j} ||t|| \right)$, and the value $space(j, G_{\mathbf{r}})$ as the value $2^{MaxLevel(G_{\mathbf{r}}, j)} \cdot |VT(G_{\mathbf{r}}, j)| \cdot \log_2(MaxCount(G_{\mathbf{r}}, j))$. Let $MaxSpace(G_{\mathbf{r}}) = \max_{j \in J(G_{\mathbf{r}})} space(j, G_{\mathbf{r}})$. We will see that *EdgeWiseMin* is applicable to our instance, if we have $\mathcal{O}(MaxSpace(G_{\mathbf{r}})^2)$ bits for computing it. The problem with $MaxSpace(G_{\mathbf{r}})$ is that it is exponential in $MaxLevel(G_{\mathbf{r}}, j)$, but this value depends on the maximum number of tuples that shares the same values for attributes VT and J and differs on at least one among the attributes X, Y, and Z in the original instance $\mathbf{r}$. As we say above, we assume this value to be manageable as it happens in many real world applications. Hereinafter, we will suppose to have $\mathcal{O}(\mathit{MaxSpace}(G_{\mathbf{r}})^2)$ bits for performing our computation.



V'	$\mathcal{W}$'(source,V')	$\mathcal{W}$'(V',sink)	V'	$\mathcal{W}$'(source,V')	$\mathcal{W}$'(V',sink)	V'	$\mathcal{W}$'(source,V')	$\mathcal{W}$'(V',sink)
$\{v_1\}$	0	10	$\{v_6\}$	5	5	$\{v_{11}\}$	10	0
$\{v_2\}$	2	8	$\{v_7\}$	7	3	$\{v_2, v_3\}$	1	8
$\{v_3\}$	2	8	$\{v_8\}$	8	2	$\{v_4, v_5\}$	3	6
$\{v_4\}$	4	6	$\{v_9\}$	7	3	$\{v_7, v_9\}$	6	3
$\{v_5\}$	4	6	$\{v_{10}\}$	10	0	$\{v_{10}, v_{11}\}$	9	0

Figure 3-14: The unfolding of the *wL-DAG* of Figure 3-5 into a weighted DAG for solving the MAX-TP problem. The table below the graph provides the weights for source-to-node edges and node-to-sink edges, which are both represented by dashed lines. Continuous edges without labels have weight 0. $P = \mathit{source}\{v_1\}\{v_2, v_3\}\{v_7, v_9\}\{v_8\}\{v_{11}\}\mathit{sink}$ is a source-sink shortest path with value 4.

Let us suppose to have a *wL-DAG* $\mathcal{L}_{\mathcal{G}} = (V, E, \mathcal{L}, \mathcal{W})$ and we want to solve MAX-TP on it. Given a *wL-DAG* $\mathcal{L}_{\mathcal{G}} = (V, E, \mathcal{L}, \mathcal{W})$, a subset $V' \subseteq V$ is a *level-subset* if and only if $V' \neq \emptyset$ and $\mathcal{L}(v) = \mathcal{L}(v')$ for every $v, v' \in V'$. Let $\mathcal{V} = \{V' \subseteq V : V' \text{ is a level-subset}\}$. By definition of level subset the function $\mathcal{L}' : \mathcal{V} \rightarrow \mathbb{N}$, such that for every $V' \in \mathcal{V}$ we have $\mathcal{L}'(V') = \mathcal{L}(v)$ for some $v \in V'$, turns out to be well defined. We define the unfolding of *wL-DAG* $\mathcal{L}_{\mathcal{G}} = (V, E, \mathcal{L}, \mathcal{W})$ as the weighted DAG $U(\mathcal{L}_{\mathcal{G}}) = (\mathcal{V} \cup \{\mathit{source}, \mathit{sink}\}, E', \mathcal{W}_{E'})$, where

$$E' = \begin{array}{c} \{(source, V') : V' \in \mathcal{V}\} \cup \{(V', sink) : V' \in \mathcal{V}\} \cup \\ \{(V', V'') \in \mathcal{V} \times \mathcal{V} : \forall v \forall v' (v \in V' \land v' \in V'' \rightarrow (v, v') \in E)\} \end{array},$$

For every $(source, V') \in E'$ we have

$$\mathcal{W}'(source, V') = \sum_{v \in V \setminus V' : \mathcal{L}(v) \leq \mathcal{L}(V')} \mathcal{W}(v),$$

For every $(V', sink) \in E'$ we have

$$\mathcal{W}'(V', sink) = \sum_{v \in V \mathcal{L}'(V') < \mathcal{L}(v)} \mathcal{W}(v),$$

And for every $(V', V'') \in E'$ with $V', V'' \in \mathcal{V}$ we have

$$\mathcal{W}'(V', V'') = \sum_{v \in V \setminus V'' : \mathcal{L}'(V') < \mathcal{L}(v) \mathcal{L}'(V'')} \mathcal{W}(v).$$

For instance, the DAG in Figure 3-14 is the result of unfolding the *wL-DAG* in Figure 3-5. The unfolding of a *wL-DAG*, in the worst case scenario, is exponential in the size of $\mathcal{L}_{\mathcal{G}}$. Given a *wL-DAG* $\mathcal{L}_{\mathcal{G}} = (V, E, \mathcal{L}, \mathcal{W})$, we define $W_{all}(\mathcal{L}_{\mathcal{G}}) = \sum_{v \in V} \mathcal{W}(v)$ as the sum of all the weights associated to nodes in V. It is straightforward to prove that the union of all the internal nodes in a *source*-to-*sink* path in the unfolding of a *wL-DAG* $\mathcal{L}_{\mathcal{G}}$ is a thick path in $\mathcal{L}_{\mathcal{G}}$ and, on the other hand, every thick path in $\mathcal{L}_{\mathcal{G}}$ may be associated with a *source*-to-*sink* path in its unfolding. Moreover, for every *source*-to-*sink* path p in the unfolding of $\mathcal{L}_{\mathcal{G}}$, let w_p be its weight. The weight of the thick path associated with p is exactly $W_{all}(\mathcal{L}_{\mathcal{G}}) - w_p$ (i.e., $\mathcal{L}_{\mathcal{G}}$ admits a thick path with value $W_{all}(\mathcal{L}_{\mathcal{G}}) - w_p$). With these premises we can prove the following result.

Theorem 4. Given a *wL-DAG* $\mathcal{L}_{\mathcal{G}} = (V, E, \mathcal{L}, \mathcal{W})$, let w the value of the shortest *source*-to-*sink* path in its unfolding. We have that the value of MAX-TP on $\mathcal{L}_{\mathcal{G}} = (V, E, \mathcal{L}, \mathcal{W})$ is equal to $W_{all} - w$.

Given a color-partial solution $CPS = (G_{\mathbf{r}}, \mathcal{C}^+, \mathcal{C}^-)$, procedure *EdgeWise* computes the value $val(CPS)$ (performed by procedure *PartialSolution* in Figure 3-13) summing up all the values M^j_{CPS} for every $j \in J(G_{\mathbf{r}})$. Each M^j_{CPS} is computed as value of a *source*-to-*sink* shortest path (performed by procedure *SourceSinkShortestPath* in Figure 3-11) on the unfolding of $\mathcal{L}^j_{CPS}$ (built by procedure *BuildDag* in Figure 3-11). For building $U(\mathcal{L}^j_{CPS})$, on which we will compute value of a *source*-to-*sink* shortest path, we may need $\mathcal{O}(MaxSpace(G_{\mathbf{r}})^2)$ bits.

Finally, let us observe that procedure *PartialSolution* does not return only the value $val(CPS)$ of the current color-partial solution $CPS = (G_{\mathbf{r}}, \mathcal{C}^+, \mathcal{C}^-)$. Since it effectively computes a minimal solution $(G_{\mathbf{r}}, G^+_{\mathbf{r}}, G^-_{\mathbf{r}})$ induced by CPS in order to provide $val(CPS)$, it returns the set $colors(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G^-_{\mathbf{r}}, G^-_{\mathbf{r}})$, that is, the set of all and only the colors associated to active edges in $(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G_{\mathbf{r}}^{-}, G_{\mathbf{r}}^{-})$. If such a set does not contain two colors (x, y, z) and (x, y, z') such that $z \neq z'$, then we have that $(G_{\mathbf{r}}, \mathcal{C}^{+}, \mathcal{C}^{-})$ is a consistent solution and we may return $val(CPS)$. Otherwise, procedure *EdgeWiseMin* takes a color (x, y, z) in $colors(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G_{\mathbf{r}}^{-}, G_{\mathbf{r}}^{-})$, such that there exists (x, y, z') in $colors(G_{\mathbf{r}}, G_{\mathbf{r}} \setminus G_{\mathbf{r}}^{-}, G_{\mathbf{r}}^{-})$ with $z \neq z'$ and performs two recursive calls, one in which $\mathcal{C}^{+}$ is updated to $\mathcal{C}^{+} \cup \{(x, y, z)\}$ and the other in which $\mathcal{C}^{-}$ is updated to $\mathcal{C}^{-} \cup \{(x, y, z)\}$.

3.4 Mining *APE*-FDS

In this section, we consider the problem of mining *APE*-FDs on a given instance $\mathbf{r}$ of a temporal schema R from a practical point of view. We will describe a prototype that performs such task. In particular, we point out two big computational challenges we addressed in the implementation of our prototype.

Let us start with the formal definition of our problem. Given a temporal schema $R = U \cup \{VT\}$, an instance $\mathbf{r}$ of R, a non-empty set $J \subseteq U$, a threshold $0 \leq \epsilon \leq 1$, and a value $k \in \mathbb{N}$, we denote as $\mathcal{PE}(\mathbf{r}, k, \epsilon)$ the set of all the *APE*-FDs $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$, formally introduced in Section 3.1, such that $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$.

A set $\mathcal{S}$ of *APE*-FDs is *complete* if and only if for every $[\Delta_k(\tau_J^R)]\ X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ belonging to $\mathcal{S}$, there exists $X' \subseteq X$ and $\overline{Y}' \subseteq \overline{Y}$ such that $[\Delta_k(\tau_J^R)]\ X'\overline{Y}' \overset{\varepsilon}{\to} \overline{Z} \in \mathcal{PE}(\mathbf{r}, k, \epsilon)$. A complete *APE*-FD-set $\mathcal{PE}(\mathbf{r}, k, \epsilon)$ is *minimal* if and only if for every $[\Delta_k(\tau_J^R)]\ X\overline{Y} \overset{\varepsilon}{\to} \overline{Z} \in \mathcal{S}$, set $\mathcal{S} \setminus \{[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}\}$ is not complete anymore. Given a complete minimal *APE*-FD-set $\mathcal{PE}(\mathbf{r}, k, \epsilon)$, every subset $\overline{\mathcal{PE}(\mathbf{r}, k, \epsilon)} \subseteq \mathcal{PE}(\mathbf{r}, k, \epsilon)$ is called a *minimal partial*.

Given a temporal schema $R = U \cup \{VT\}$, an instance $\mathbf{r}$ of R, a non-empty set $J \subseteq U$, a threshold $0 \leq \epsilon \leq 1$, and a value $k \in \mathbb{N}$, we are interested in finding a minimal complete set $\mathcal{PE}(\mathbf{r}, k, \epsilon)$. However, in order to do that we have to deal with two computational problems:

- there exist temporal schemas $R = U \cup \{VT\}$, instances $\mathbf{r}$ of R, non-empty sets $J \subseteq U$, thresholds $0 \leq \epsilon \leq 1$, and values $k \in \mathbb{N}$, for which the smallest minimal complete set $\mathcal{S}$ w.r.t. such parameters is exponential in the size of U;

- given a single *APE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ on a schema $R = U \cup \{VT\}$, and an instance $\mathbf{r}$ of R, deciding whether or not $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ is a NP-complete problem.

The first result may be obtained by adapting a result of Kivinen et al. [62] on approximate functional dependencies. The second result is proved in Appendix A. However, such theoretical bounds are both difficult to achieve in real world domains. For instance, the size of $\mathcal{PE}(\mathbf{r}, k, \epsilon)$ could be exponential in $|U|$, but in real case scenarios we have that $|U|$ is less or equal than 50 elements. Moreover, the instance built in [62] for achieving the exponential lower bound is far from real world instances. An even worse situation occurs for the complexity of checking a single *APE*-FD: such a problem is NP-Complete in the number of tuples and thus we cannot rely on the fact that such set is small. Quite the opposite in fact, since instances are supposed to grow day by day. This problem is known as the *curse of cardinality*, whose relevance has been recently pointed out for temporal inference of sequential patterns in [68]. As in the previous case, we observe that the instance specified in Appendix A for proving the NP-hardness result has been built in a very complex and constrained way. Such an instance does not even remotely resemble some real world scenario. Thus, we are allowed to design and implement a prototype for the practical mining of such dependencies on real world datasets, and evaluate its performances.

3.4.1 Prototype Overview

Not discouraged by the dramatic premise reported in Appendix A, we developed a prototype that, given a temporal schema $R = U \cup \{VT\}$, an instance $\mathbf{r}$ of R, a non-empty set $J \subseteq U$, a threshold $0 \leq \epsilon \leq 1$, and a value $k \in \mathbb{N}$ returns a minimal complete set $\mathcal{PE}(\mathbf{r}, k, \epsilon)$. The prototype is called $\mathcal{A}$*ttila*, which stands for *Approximate Temporal Tailored Inference Lean Application*. In the following, we provide a high-level description of $\mathcal{A}$*ttila* modules and their interaction. Moreover, we provide a detailed description of novel ideas underlying key features of the prototype. The implementation of $\mathcal{A}$*ttila* follows the principles of distributed programming, where different tasks are executed by different processes (possibly executed on different machines). $\mathcal{A}$*ttila* is composed by three main processes:

- Worker is responsible for keeping the current status by maintaining a representation of the minimal partial $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ of $\mathcal{PE}(\mathbf{r}, k, \epsilon)$. Worker maintains a compact representation of the set of the APE-FDs checked so far as well as a compact representation of the APE-FDs that remain to be checked in the future. Worker updates its state according to the last APE-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ that has been checked. The next state depends on the fact that either $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ or $\mathbf{r} \not\models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$. In both cases Worker marks $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ to not be checked anymore. In this way, it can determine precisely the next APE-FD to check;

- Contributor is responsible for checking a single APE-FD against the instance $\mathbf{r}$. Contributor retrieves APE-FD from Worker. When Contributor gets an APE-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$, it schedules jobs for checking $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ among the computational units that will effectively check the given APE-FD;

- Sub-Contributor is the basic computational unit that depends on Contributor. Sub-Contributor takes advantage of the graph representation generated by Contributor for resolving a sub-problem of the original one. More precisely, Sub-Contributor receives a sub-problem of the one stored by Contributor for checking a given $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ against it. During the computation, Contributor may send a reduction of the sub-problem Sub-Contributor is dealing with. This is due to the fact that Contributor may assign portions of a problem to several Sub-Contributors.

Processes composing $\mathcal{A}$*ttila* are hierarchically organized. We may have one Worker that manages minimal partial $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$, but more than one Contributor is needed for checking several dependencies at the time. Each Contributor may have several Sub-Contributor in order to speed-up the checking procedure.

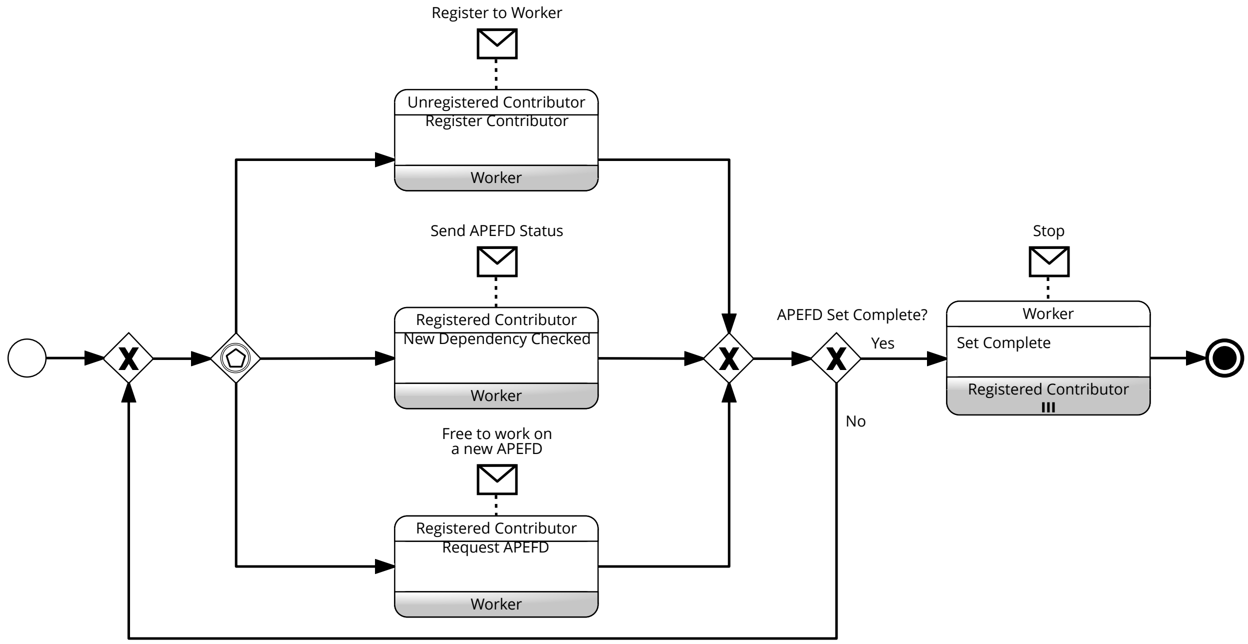


Figure 3-15: A BPMN choreography showing the interaction between a Worker and its (possibly) many Contributors.

In the following, we provide a detailed description of each process type in order to give a general idea of how computations are handled. Worker is responsible for managing the minimal partial $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$. At the end of computation, $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ turns out to be a minimal complete set, which is the goal of our distributed procedure. Worker interacts only with its pool of Contributors. A graphical account of how such interaction happens is given by the BPMN choreography [82] reported in Figure 3-15. A Contributor can register to the Worker at any time increasing by one the number of APE-FDs that may be checked simultaneously. Worker may receive APE-FD request only by registered Contributors. For this reason, it keeps two auxiliary APE-FD-sets $Pending$ and $Assigned \subseteq Pending$. $Pending$ is a minimal set of APE-FDs such that $Pending \cap \overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon) = \emptyset$ and, if for every $\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z} \in Pending$ $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$, then $Pending \cup \overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ is a minimal complete APE-FD-set. $Pending$ contains all APE-FDs among whom Worker must choose every time is asked for an APE-FD by some of its Contributor. $Assigned$ represents all APE-FDs in $Pending$ already assigned to Contributors. Every time a Contributor returns the result of APE-FD-check on an element of $Assigned$, $Pending$ and $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ are updated according to the fact that either $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ or $\mathbf{r} \not\models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$. More precisely, we have that $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ is removed from both $Pending$ and $Assigned$, and

- if $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$, then Worker inserts such APE-FD in $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$;

- if $\mathbf{r} \not\models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$, then for each attribute $W \in U \setminus (X \cup Y \cup \{Z\})$, dependencies $[\Delta_k(\tau_J^R)]XW\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ and $X\overline{YW} \xrightarrow{\varepsilon} \overline{Z}$ are inserted in $Pending$.

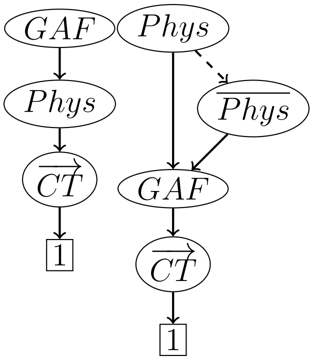


Figure 3-16: The update of the set $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon) = \{[\Delta_k(\tau_{PatId}^{Contact})]GAF, Phys \xrightarrow{\varepsilon} \overline{CT}\}$ (left) into the set $\overline{\mathcal{PE}}'(\mathbf{r}, k, \epsilon) = \{[\Delta_k(\tau_{PatId}^{Contact})]GAF, Phys \xrightarrow{\varepsilon} \overline{CT}, [\Delta_k(\tau_{PatId}^{Contact})]GAF, \overline{Phys} \xrightarrow{\varepsilon} \overline{CT}\}$ (right).

Basically Worker performs three operations: (i) it extracts an *APE*-FD $[\Delta_k(\tau_J^R)]$ $X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ from *Pending*, (ii) it updates $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ when it is the case, and (iii) it updates *Pending* and *Assigned* when the status of a new dependency is discovered. In order to efficiently perform such operations we make use of *Ordered Binary Decision Diagrams* (OBDDs)[21]. An OBDD is a single-rooted directed acyclic graph representing a propositional formula ψ. Every node is labeled with a propositional variable, except for the only terminal 1[1]. From every non-terminal node v we have at most two outgoing edges $low(v)$ (dotted line, as in Figure 3-16) and $high(v)$ (solid line, as in Figure 3-16), where $low(v)$ means that variable v is taken with value 0, while $high(v)$ means that such a variable is taken with value 1. Every path from the root to terminal node 1 represents a variable truth assignment. Thus, an OBDD may be seen as the set of all truth assignments for some formula ψ. Worker keeps track of sets $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$, *Pending*, and *Assigned* by means of three formulas $\psi_{\overline{\mathcal{PE}}}, \psi_P$ and ψ_A, represented by different OBDDs, respectively. The *APE*-FDs in $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ (resp. in *Pending* and *Assigned*) represent all and only the solutions of $\psi_{\overline{\mathcal{PE}}}$ (resp. of ψ_P and of ψ_A). Hereinafter, with a little abuse of notation we will use ψ for denoting both the formula and the OBDD that represents all its possible solutions. Informally, updates of these three sets are implemented by adding conjuncts/disjuncts to their respective formulas.

For representing set $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ as all and only the solutions of a formula, it suffices to assign to each attribute $X_i \in U$ three variables $x_i, \overline{x}_i, \overrightarrow{x}_i$. Then, formula $\psi_{\overline{\mathcal{PE}}}$ has be satisfied by assignments $\sigma : \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n, \overrightarrow{x}_i, \ldots, \overrightarrow{x}_n\} \to \{0, 1\}$. An *APE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ belongs to $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ if and only if the assignment $\sigma(x_i) = 1 \;\forall\; X_i \in X$, $\sigma(\overline{y}_j) = 1 \;\forall\; \overline{Y}_j \in \overline{Y}$, and $\sigma(\overrightarrow{z}) = 1$ satisfies $\psi_{\overline{\mathcal{PE}}}$. According to this approach, for instance, $a_1 \wedge a_2 \wedge \overline{a}_3 \wedge \overrightarrow{a}_4$ represents *APE*-FD $[\Delta_k(\tau_J^R)]A_1A_2\overline{A}_3 \to \overline{A}_4$. Clearly, if a formula $\psi_{\overline{\mathcal{PE}}}$ represents all the possible *APE*-FDs, an OBDD for $\psi_{\overline{\mathcal{PE}}}$ represents them as well. The same approach may be used for sets *Pending* and *Assigned*. Hereinafter, we refer to ψ as the OBDD representing all and only the assignments σ such that $\sigma \models \psi$.

A Worker begins the *APE*-FD mining task by initializing two OBBDs representing $\psi_{\overline{\mathcal{PE}}} =$

[1] We use OBDD without the 0 terminal.

$\psi_A = \bot$. Initially, $\psi_{\overline{\mathcal{PE}}} = \bot$ (i.e., $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon) = \emptyset$) since the distributed procedure has not discovered any valid APE-FD yet. ψ_P is true only for those assignments that represent well-formed APE-FDs. In order to extract from ψ_P an APE-FD to be tested, we take the solution associated with any root-to-terminal path in the OBDD for $\psi_P \wedge \neg\psi_A$. For inserting an APE-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ in $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ it suffices to update $\psi_{\overline{\mathcal{PE}}}$ to $\psi_{\overline{\mathcal{PE}}} \vee \psi$ where $\psi = \bigwedge_{X_i \in X} x_i \wedge \bigwedge_{X_i \in Y} \overline{x}_i \wedge \overrightarrow{z}$. Moreover, for deleting a solution from *Pending* it suffices to update ψ_P:

- if we have that $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$, then we update ψ_P to $\psi_P \wedge \neg\psi$;
- if we have that $\mathbf{r} \not\models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$, then we update ψ_P to $\psi_P \wedge \overline{\psi}$ where $\overline{\psi} = \overrightarrow{z} \to (\bigvee_{X_i \in U \setminus X} x_i \vee \bigvee_{X_i \in U \setminus Y} \overline{x}_i)$ (i.e., if $\overrightarrow{z}$ holds, then at least one of variables not contained in the formula for the given APE-FD must hold).

It is worth noting that we have a search operation for APE-FDs that is linear in $|U|$. Moreover, boolean operations on OBDDs are implemented in very efficient way by many packages on the market (in our prototype we used BuDDy [67]). This solution allows us to have a compact representation of sets of APE-FDs that can be manipulated efficiently.

Figure 3-16 shows an example of how $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ is updated if we represent it through formula $\psi_{\overline{\mathcal{PE}}}$. In this example, we borrow two dependencies from the psychiatric case register introduced in a simplified way in Section 3.1 and discussed in detail in Section 3.4.2. The real world schema differs from the example since patients are identified by *PatId* attribute in place of their names. Furthermore, several attributes are used for storing information regarding registered calls. The most significant attribute is *Global Assessment of Functioning* (*GAF*): it is a numeric value provided by the physician at the end of the call, and it scores the patient's mental health status. Figure 3-16 shows an update operation on $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon) = \{[\Delta_k(\tau_{PatId}^{Contact})]GAF, Phys \overset{\varepsilon}{\to} \overline{CT}\}$, where $\overline{\mathcal{PE}}(\mathbf{r}, k, \epsilon)$ is updated to $\overline{\mathcal{PE}}'(\mathbf{r}, k, \epsilon) = \{[\Delta_k(\tau_{PatId}^{Contact})]GAF, Phys \overset{\varepsilon}{\to} \overline{CT}, [\Delta_k(\tau_{PatId}^{Contact})]GAF, \overline{Phys} \overset{\varepsilon}{\to} \overline{CT}\}$. As one may notice, each node v has at most two outgoing edges, one solid and one dashed representing $high(v)$ and $low(v)$ respectively. Taking a solid edge $high(v)$ in a root-to-terminal path denotes that the attribute corresponding to v belongs to the dependency. Taking a dashed edge $low(v)$ in a root-to-terminal path denotes that the attribute corresponding to v does not belong to the dependency. As an example, the OBDD shown in Figure 3-16 (right) features two paths from the root node, labeled $Phys$, to terminal node 1. Since the edge $(Phys, \overline{Phys})$ is dashed, the path $Phys, \overline{Phys}, GAF, \overrightarrow{CT}, 1$ represents the dependency $[\Delta_{+\infty}(\tau_{PatId}^{Contact})]GAF, \overline{Phys} \overset{\varepsilon}{\to} \overline{CT}$. Such dashed edge implies that attribute $Phys$ is not taken in X while it is taken in Y because the outgoing edge the path takes from $\overline{Phys}$ is a continuous one.

Let us consider now the Contributor process. Until now we just considered it as a process which asks Worker for an dependency and eventually answers whether $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \overset{\varepsilon}{\to} \overline{Z}$ holds or not, as shown by the BPMN choreography in Figure 3-15. Thus, Contributor is responsible for checking a single APE-FD at a time. Since the complexity is intractable (recall that the problem in NP-Complete), Contributor does not deal directly with the computation. Indeed, as mentioned before, it splits a problem among several computational units called *Sub-Contributors*.

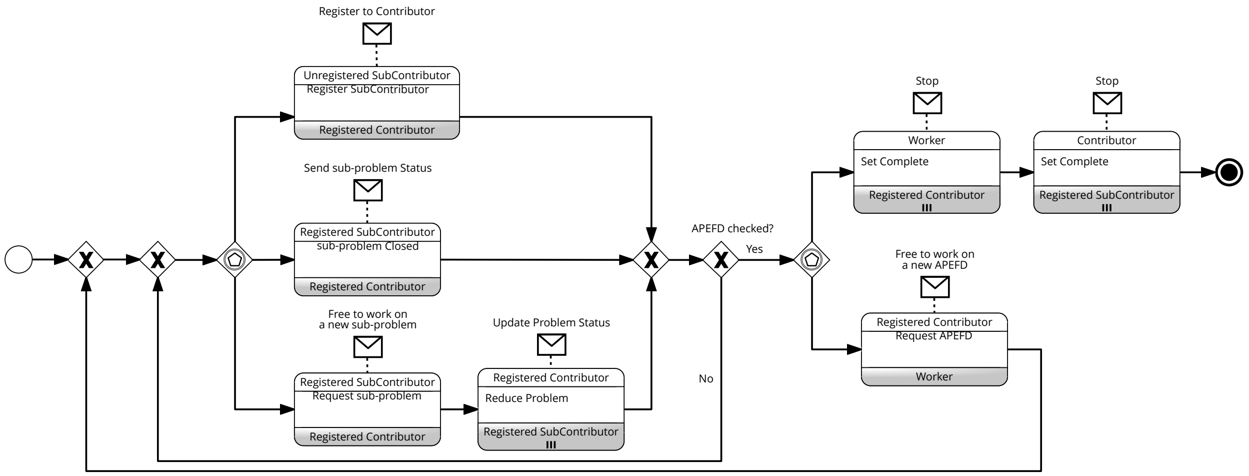


Figure 3-17: A BPMN choreography showing the interaction between a Contributor and its (possibly) many Sub-Contributors.

The way in which a Contributor deals with its pool of Sub-Contributors is very close to the interaction between Worker and its Contributors and it is described by the BPMN choreography diagram provided in Figure 3-17. The status of the problem is managed by Contributor and it is represented by a binary tree, where each node is labelled with a tuple and its two children represent either the case in which the tuple is inserted in the current solution or it is removed form it. Sub-problems are generated by asserting that a tuple belongs or not to the final solution. This procedure generates a tree. Initially the whole tree is given to a single Sub-Contributor to visit. Suppose that a new Sub-Contributor registers himself to the same Contributor during a computation. Such Contributor selects the sub-problem of a Sub-Contributor and a tuple t in $\mathbf{r}$. Contributor splits the sub-problem in two parts, one in which t must belong to the solution and the other in which t does not belong to the solution. One portion is given to the new Sub-Contributor and the "old" Sub-Contributor is notified to reduce its problem to the other portion. Usually we have multiple Sub-Contributors that work in a subtree rooted at the node where the reduce operation happens, and thus, as reported in Figure 3-17, we have to notify all of them about the reduction.

Figure 3-18 shows an example of how Contributor works. Suppose that there is exactly one Sub-Contributor sc_1 that is exploring the tree (a) on the top of Figure 3-18. At a certain point, a new Sub-Contributor sc_2 registers himself to the Contributor and requests a sub-problem. Now Contributor looks at the active sub-problem and chooses the one of sc_1. At the root of such problem there is tuple t_1. Therefore, Contributor splits the sub-problem in two more sub-problems: one where t_1 is forced to be deleted (tree (b) in Figure 3-18), and the other where t_1 is kept (tree (c) in Figure 3-18). Finally, the exploration of sub-tree (c) is given to sc_2 and sc_1 is notified that its exploration of the tree (a) is reduced to the exploration of the subtree (b).

Sub-Contributor is the minimal computation unit: it simply performs tasks assigned by its master Contributor. Sub-Contributor listens constantly to its Contributor in order to receive

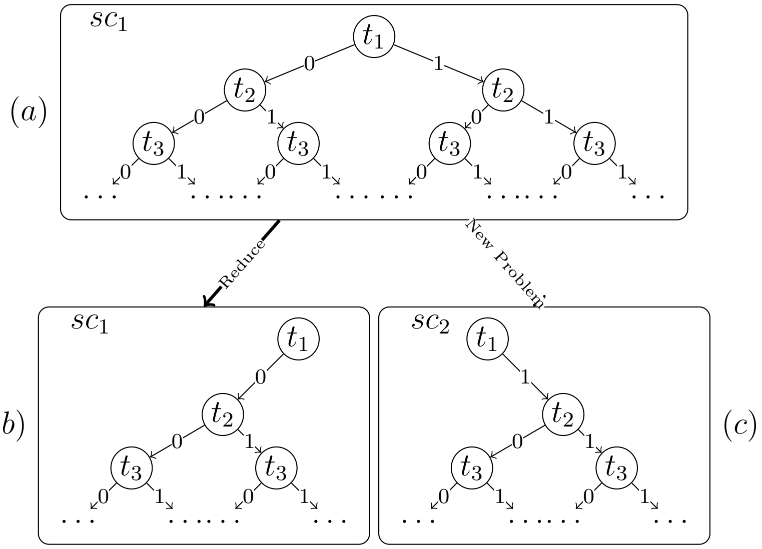


Figure 3-18: A graphical account of how the tree is splitted among Sub-Contributors.

reduction of its current sub-problem for speeding up the process, meanwhile it explores its current sub-problem searching for a solution. Sub-Contributor operates in two symmetric ways that may be seen as two concurrent threads. The first thread assumes that its sub-problem contains the solution and performs a depth-first search of the tree in order to find it. The other thread assumes that the sub-problem does not contain the solution and tries to find a counter-example. In order to deal with the latter task, the Sub-Contributor translates the sub-problem into a Linear Programming problem and verifies its feasibility. This symmetric approach turns out to be very efficient. *Attila* makes use of the open-source linear programming library *GNU Linear Programming Kit* (GLPK) [89] to perform such linear programming tasks.

3.4.2 Mining *APE*-FDs on Clinical Domains

In the following, we discuss *APE*-FDs extracted by means of the introduced prototype. These results provide also an early validation of our prototype. In particular, we considered two application domains. The first one refers to *psychiatry*. Data regarding contacts between patients and psychiatrists are collected. Section 2.3.2 provides a detailed description of this domain. We extracted from relation *Contact* for the psychiatric domain the following *APE*-FDs:

- $[\Delta_{+\infty}(\tau^{Contact}_{PatId})]GAF \overset{\varepsilon}{\to} \overline{numPsychologists}$ with $\epsilon = 0.1$. This dependency states that, for each pair of consecutive calls of the same patient, the number of psychologists of the second call uniquely depends on patient's GAF score of the first call. This may indicate that some (even implicit) policy determines the number of psychologists needed for following patients, according to the previous patients' condition.

- $[\Delta_{+\infty}(\tau^{Contact}_{PatId})]Service, GAF \overset{\varepsilon}{\to} \overline{CT}$ with $\epsilon = 0.1$. This dependency states that, for each pair of consecutive calls for the same patient, the second contact type (family member, a neighbour, the police, . . .) uniquely depends on the previous patient's GAF score and Service (clinical psychiatry, medical psychology, psychotherapy, . . .).

- $[\Delta_{+\infty}(\tau^{Contact}_{PatId})]GAF, Physician \xrightarrow{\varepsilon} \overline{Request}$ with $\epsilon = 0.1$. This dependency states that, for each pair of consecutive calls of the same patient, the request (it could be group psychotherapy, family psychotherapy, legal medical evaluation, . . .) of the second call depends on the physician and on the patient's GAF score of the previous call.

The other domain we considered is the *pharmacovigilance* one, and it is described in Section 2.3.3. The temporal data are used to investigate any cause-effect relationship among drugs and reaction(s) in different time periods, or according to the exposure time frame. As for the temporal feature, we focused here on the evolution of reports considering single drugs (by using *PhProd* (Pharmaceutical Product, i.e., active principle) as join attribute in the evolution expression). Indeed, we could investigate physician behaviors through time w.r.t. previously reported adverse events. In other words, it would be interesting to verify if past adverse event cases lead to future good practices. As an example, changes in drug dosages could be related to the fact that physicians are aware of past cases of adverse events. Reducing or increasing dosages of previously suspected drugs may be seen as attempt to avoid such adverse events.

Among *APE*-FDs extracted from a recent instance of *Reports* schema of the Italian Network of Pharmacovigilance we introduce here the following *APE*-FD.

- $[\Delta_{+\infty}(\tau^{Reports}_{PhProd})]PhProd, Dos \xrightarrow{\varepsilon} \overline{Dos}$ with $\epsilon = 0.2$. Such a dependency may point out that, when an ADR of a drug is reported, the dose is usually adjusted in the same way, depending on the previous administrations to possibly different patients. This is a good indicator that Italian physicians methodically use the Italian Network of Pharmacovigilance in managing patients' therapies.

3.4.3 Performance Analysis

In this section, we provide a short and preliminary performance analysis of the various components of *Attila*. We did run two kinds of test. The first one was on a single machine, to have a first estimate of the time required for mining *APE*-FDs on a large real-world database. The second test involved a single *APE*-FD, checked using a server and at most two distinct remote machines. By this test we wanted to observe how significantly the time required for checking a single *APE*-FD decreases, when the problem is split among different computational units.

First of all, we analyzed performances regarding the whole system. In particular, we considered the case when the computation is entirely done by only a physical machine. We tested our prototype on an instance of *Contact* schema. It consists of approximately $1.5 \cdot 10^6$ rows. *APE*-FDs of the kind $[\Delta_{+\infty}(\tau^{Contact}_{PatId})]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ were extracted with $\epsilon = 0.1$. Figure 3-19 shows the outcome of such experiment. *Attila* tested almost 4500 *APE*-FDs in approximately 10 days. Figure 3-19 shows that among tested *APE*-FDs (i.e., holding and not holding), a large portion of *APE*-FDs, denoted by *superset*, is subsumed by holding *APE*-FDs.

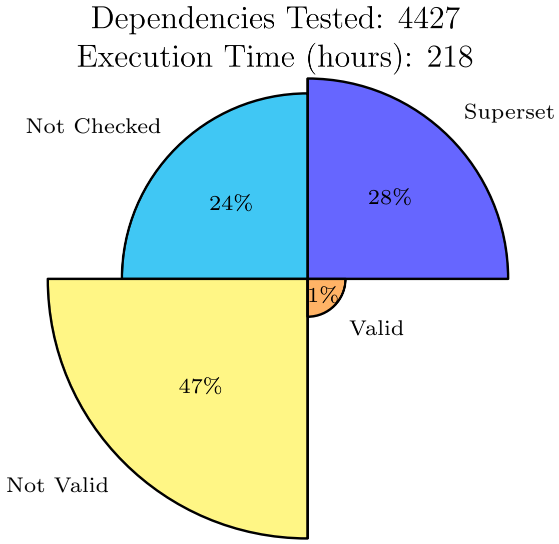


Figure 3-19: The result of the execution of $\mathcal{A}ttila$ on a single machine (Intel Core i3(TM) CPU M 330 2.13GHz, 4GB) on an instance of table *Contact* ($\sim 1.5 \cdot 10^6$ rows).

Thus, they have not been tested (i.e., only minimal *APE*-FDs need to be tested). Worker is often idling, since most of the computation is performed by Contributors. As discussed in Section 3.4, a Contributor must visit a tree, which is exponentially large in the instance size. This operation is theoretically unfeasible for large instances. However, by employing simple pruning conditions (e.g., too many tuples deleted, violated constraints, and so on), the tree size may be decreased.

We analyzed the interactions between Contributor and Sub-Contributor by using *APE*-FD $[\Delta_{+\infty}(\tau^{Reports}_{PhProd})]PhProd, Dos \xrightarrow{\varepsilon} \overline{Dos}$ with $\epsilon = 0.2$, as described in Section 3.4.2. Figure 3-20 shows comparison between various configurations of Sub-Contributors when checking *APE*-FDs. We considered five possible situations:

(i) a single local Sub-Contributor (Server) running on the Contributor machine;

(i) a single local Sub-Contributor, and a remote Sub-Contributor (Remote);

(i) two local Sub-Contributors and a single remote Sub-Contributor,

(i) two remote Sub-Contributors;

(i) a single local Sub-Contributor, and two remote Sub-Contributors.

By two remote Sub-Contributors we intend two separate physical machines with identical hardware/specs running a Sub-Contributor each. As expected, the performances improve when the task is distributed among different machines.

Figure 3-21 depicts the number of closed branches, which increases according to the size of the instance. We can conclude that our instances are easier to solve than the fictional instances used to prove NP-hardness results.

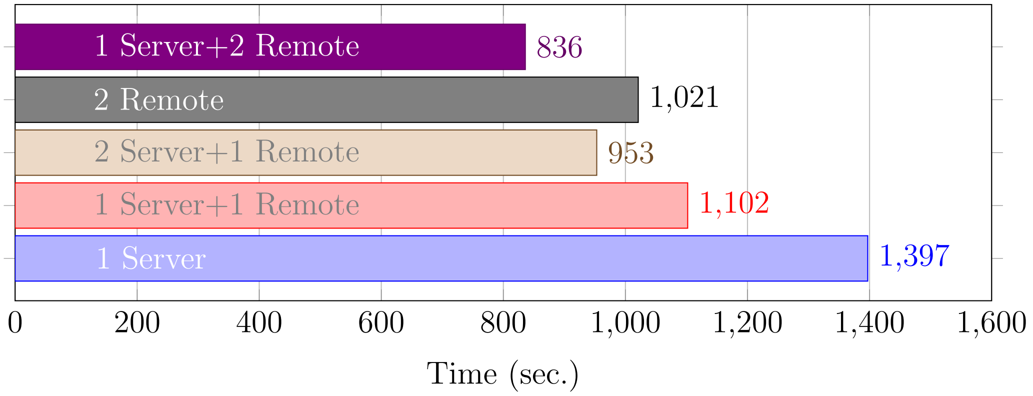


Figure 3-20: The result of *Attila* execution varying Sub-Contributors configurations on the same instance of the schema *Reports* consisting of $\sim 1.5 * 10^5$ rows (Server: 6 Core AMD Opteron(TM) 4284 3GHz, 8GB, Remote: AMD Phenom(TM) II X6 1055T Processor 2.8 GHz, 8GB).

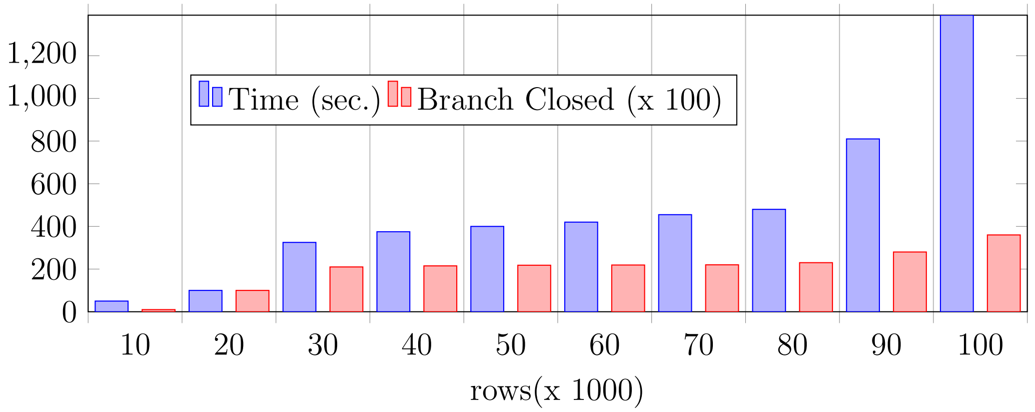


Figure 3-21: The number of closed branches considering incremental portions of the same instance of schema *Reports* (the time refers to the execution on a Intel Core i3(TM) CPU M 330 2.13GHz, 4GB machine).

3.5 Conclusions

In this chapter, we proposed a framework for extracting *Approximate Pure Temporally Evolving Functional Dependencies* (*APE*-FDs for short) from a temporal database. Moreover, we have addressed the data complexity of such problem. Unfortunately this complexity turns out to be NP-Complete even for the most simple of such dependencies. When we look for the set of *APE*-FDs holding over an instance **r**, sizes of the result set are determined by the number of attributes. For some instances, the lower bound of such size is exponential. We dealt with these problems by using model checking techniques, distributed computations, and linear programming techniques. The implemented prototype has proved to be efficient in two real world clinical scenarios. Moreover, we discussed interesting *APE*-FDs extracted from psychiatry and pharmacovigilance domains. These results may provide analysts with a better understanding of the underlying data. We plan to further improve and extend our prototype in order to deal with different types of approximate functional dependencies. Moreover, we plan to further validate mined *APE*-FDs with clinical experts.

Chapter 4

Discovering Quantitative Temporal Functional Dependencies on Clinical Data

In the previous chapter, we proposed a new kind of functional dependency (*APE*-FD), based on *Pure Temporally Evolving* that is one particular class of *T*-FDs (as illustrated in detail in Section 2.1.2). In this chapter, we focus on a new kind of approximate temporal functional dependency that uses *Pure Temporally Grouping.* This functional dependency is more focused on the *quantitative* aspect of the data. In the original concept of *AT*-FDs, shown in Section 2.1.4, the comparisons between atemporal attributes are done by using only equality. For this reason, if we want to analyze data from an OLAP perspective, such dependencies are not well suited for extracting rules related to some quantitative data, in particular when we want to analyze different values for clinical measurements that have meaningless variations within some specified threshold.

To this end, we propose a new kind of approximate temporal functional dependencies, called Multi Approximate Temporal Functional Dependency (*MAT*-FD), that considers "small variations" for the values of quantitative attributes, while retaining the compactness and the simplicity of *AT*-FDs. Another novel aspect presented in this chapter is represented by the design of a mining algorithm for extracting *MAT*-FDs from a given clinical data set. In order to extract knowledge from the data set, the algorithm generates and tests possible candidates of valid *MAT*-FDs. Then, it collects the valid ones, according to some optimality criteria, both for the temporal dimension and for the involved attributes. This means that checking if the data set satisfies a given *MAT*-FDs is an operation that is intensively used during the mining process, and its complexity deeply affects the overall complexity of the whole mining process. We propose a graph-based solution for performing such operation, and we address its complexity. Finally, we develop a tool, called SW-MATFDminer, that implements the algorithm above. The tool has been designed to extract *MAT*-FDs regardless of the application domain (i.e., it is a general-purpose tool). In particular, we provide results from the use of our tool to mine data coming from intensive care units and collected within the MIMIC III dataset (that is illustrated in Section 2.3.1). Let us recall that this work has been published in [30].

4.1 Multi Approximate Temporal Functional Dependencies

In the following, we introduce a new type of approximate temporal *FDs*, based on a multidimensional relational model, usually adopted in relational OLAP approaches [61]. Here atemporal attributes of a relational schema are distinguished in dimensional (alphanumeric) and measure-related (quantitative) attributes. More formally, let us consider relations over a relational schema R with attributes $D \cup \{M\} \cup \{VT\}$, where D represents all atemporal attributes or dimensions, M represents a quantitative attribute containing values of some measure, and VT is the attribute denoting the *valid time* of tuples.

4.1.1 A Motivating Example

Throughout this chapter we will refer to examples drawn from table 4.1. The table represents a relation containing hospitalization data of patients. The schema of such relation is composed by the following attributes: *VT* (valid time), *patient* (patient name), *Systolic blood pressure* (value of systolic blood pressure), *Drug* (name of the drug), *ICD9* (International Classification of Diseases, 9th revision) [43].

Considering a time window of one week, we can observe that if the patient is under a treatment with a drug, then the values of his pressure are stable within a specified range. On the other hand, if we consider for each patient his pressure (regardless of whether he is under a treatment), we observe that the values of blood pressure are not within the specified range for the functional dependency $Patient \rightarrow Pressure$ to hold. Such kind of dependency involving quantitative data cannot be derived/represented through a T-FD or an A-FD, as they involve only textual data, compared through equality.

In this chapter, we propose a new kind of functional dependency that allows us to mine information such as *"In most cases, patients with the same treatment have the same diastolic blood pressure within a range of 10mm/Hg, considering a maximal time window of 1 week"*.

Clinicians could be interested in the properties entailed by such functional dependency. For example, assuming that the given sliding window is the maximal for which the *FD* holds, one may observe that for hospitalizations longer than 7 days the blood pressure slowly changes. Moreover, because of the approximation (as explained in section 2.1 and subsection 4.1.2), clinicians can rely on information that cannot be retrieved by means of *FDs*.

4.1.2 Definition and Model

After having introduced *FDs*, A-FDs, and T-FDs, we now introduce the concepts of DT-FDs and MAT-FDs.

In the following, we consider relations over a relational schema R with attributes $D \cup \{M\} \cup \{VT\}$, where D represents all atemporal attributes or dimensions, M represents a quantitative attribute containing values of some measure, and VT indicates a temporal attribute. For these functional dependencies, we consider pure temporal grouping of the form $[k]X \rightarrow M$ (that is proposed in [32]), where $k \in \mathbb{N}$ is the parameter of a specific temporal grouping function $t - Group$ that is, in our case, the *sliding window (SW)* grouping.

A sliding window $SW(i, k)$ includes all the time points in interval $[i, i + k - 1]$. Thus, once the length of the SW over relation r (i.e., k in the example) has been fixed, every SW over r will feature that length. Every SW sets up a group (or *chain*) over which our functional dependency is checked.

As seen in section 2.1, T-FD and A-FD can detect functional dependencies only if values for Y attribute are equal. This is a limitation when some attribute is a measure and the difference in its values is negligible.

To successfully consider some quantitative features of clinical data, we introduce the concept of Delta Temporal Functional Dependency (DT-FD). DT-FDs are functional dependencies that focus on measures, and they allow us to constrain quantitative data with some threshold when comparing numerical values. Consequently, we introduce a new parameter, namely δ, we need to use in the specification and evaluation of the new functional dependency. Intuitively, δ is a user-defined parameter and represents the maximum distance allowed between the measures of two tuples within a sliding window, for the *FD* to hold. Formally, given a relational schema $R = D \cup \{M\} \cup \{VT\}$, a natural number k, and a real number $\delta \geq 0$, a DT-FD is an expression of the form $[k]X \underset{\delta}{\rightarrow} M$ with $X \subseteq D$. Now, we define when an instance $\mathbf{r}$ of R satisfies a given DT-FD.

VT	**Patient**	**Systolic Blood Pressure**	**Drug**	**ICD9**
2017-01-07	Michael	115	Atenolol	V553
2017-01-08	Michael	145	-	V301
2017-01-09	Michael	122	Atenolol	V660
2017-01-10	Michael	125	Atenolol	V660
2017-01-11	Michael	118	Atenolol	V553
2017-01-12	Michael	131	-	V552
2017-05-01	Michael	145	Atenolol	V553
2017-05-04	Michael	142	Atenolol	V553
2017-05-07	Michael	141	Atenolol	V553
2017-01-07	Sebastian	132	Lisinopril	V553
2017-01-08	Sebastian	135	Lisinopril	V553
2017-01-09	Sebastian	170	-	V660
2017-01-10	Sebastian	128	Lisinopril	V310
2017-01-11	Sebastian	150	-	V596
2017-01-12	Sebastian	130	Lisinopril	V310

Table 4.1: A subset of one of the tables we create to mine MAT-FDs.

Patient	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike	Mike
Day	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7	7	7
Systolic blood pressure in mmHg (M)	118	122	125	118	123	119	116	120	118	117	123	121	124	145	120	116	124	122	122	120	118

Table 4.2: Systolic blood pressure of a single patient during a week.

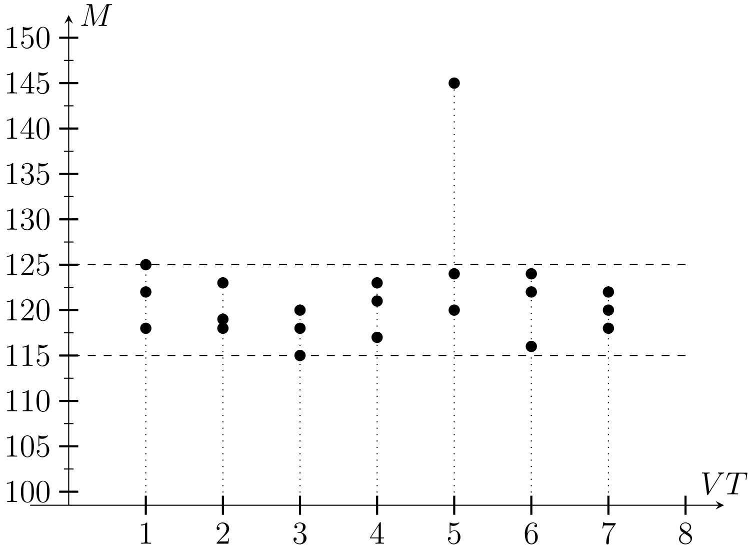


Figure 4-1: Graphic representation of data in table 4.2($\delta = 10$).

Definition 19. (DT-FD) Let $\mathbf{r}$ be an instance over relational schema $R = D \cup \{M\} \cup \{VT\}$, and $[k]X \xrightarrow[\delta]{} M$ be a DT-FD over R. We say that $\mathbf{r}$ satisfies $[k]X \xrightarrow[\delta]{} M$, written $\mathbf{r} \models [k]X \xrightarrow[\delta]{} M$, if and only if every pair of tuples $t_1, t_2 \in \mathbf{r}$ satisfies the following formula:

$$\exists i (t_1[VT], t_2[VT] \in SW(i,k)) \rightarrow |t_2[M] - t_1[M]| \leq \delta.$$

Let us notice that DT-FDs may express constraints on the range of measurements for tuples whose valid times are within a given range. If we expect the data to conform to such a DT-FDs we can easily impose it as a constraint at schema level. If it is not the case (i.e., tuples in $\mathbf{r}$ are in general not constrained to respect a DT-FD), we may be interested whether or not "almost" the tuples in our instance satisfy a given DT-FD. Then we are moving our perspective from imposing constraints, derived from our previous knowledge, on the given schema to extracting knowledge from the data present in our instance. In order to do that we take advantage of the approximation concept. Approximation is defined by means of a measurement error, namely

Q, for DT-FD. Such a measurement considers the minimum number of tuples in r to be *deleted* for the DT-FD to hold. Formally, $Q([k]X \underset{\delta}{\rightarrow} M, \mathbf{r}) = |\mathbf{r}| - max\{|\mathbf{s}| \mid \mathbf{s} \subseteq \mathbf{r} \wedge \mathbf{s} \vDash X \underset{\delta}{\rightarrow} M\}$. The related *scaled measurement* q is defined as $q([k]X \underset{\delta}{\rightarrow} M, \mathbf{r}) = Q([k]X \underset{\delta}{\rightarrow} M, \mathbf{r})/|\mathbf{r}|$.

In this approximation, a user-defined threshold ε specifies the maximum error allowed, that is the percentage of tuples in the entire relation $\mathbf{r}$ to be deleted.

In the following, we consider DT-FD with approximation, obtaining a new type of approximate functional dependency called Multi Approximate Temporal Functional Dependency (MAT-FD). Given a relational schema $R = D \cup \{M\} \cup \{VT\}$, a natural number k, a real number $\delta \geq 0$, and a real number $0 \leq \varepsilon \leq 1$, a MAT-FD is an expression of the form $[k]X \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$ where $X \subseteq D$. Now, let us define when an instance $\mathbf{r}$ of R satisfies a given MAT-FD.

Definition 20. (MAT-FD) Let $\mathbf{r}$ be an instance over the relational schema $R = D \cup \{M\} \cup \{VT\}$, and $[k]X \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$ be a MAT-FD over R. We say that $\mathbf{r}$ satisfies $[k]X \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$, written $\mathbf{r} \models [k]X \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$, if and only if $q([k]X \underset{\delta}{\rightarrow} M, \mathbf{r}) \leq \varepsilon$

Let us observe data in Table 4.2 and their representation in Figure 4-1. These data represent the values of systolic blood pressure of a patient during a week, having three measurements each day. Let $\mathbf{r}$ be the instance consisting of tuples shown in Table 4.2. Let us suppose that we want to verify DT-FD $\mathbf{r} \models [7]Patient \underset{10}{\rightarrow} Pressure$. Clearly, it is not the case because the difference between the two values of 145 and of 124, both measured on the fifth day, exceeds 10. Since those tuples happen to have the same day as valid time they are grouped together by $SW(i, 7)$ and thus $\mathbf{r} \not\models [7]Patient \underset{10}{\rightarrow} Pressure$. However, it is easy to see that almost all values fit within the given delta ($\delta = 10$) and only the second measurement of the fifth day, with a value of 145, does not belong to it. This is a typical situation where the approximation helps, because *deleting* only one tuple (i.e., the one with the value of 145 on the fifth day) allows us to obtain a valid DT-FD. In this example, ε could be up to 4.76% for the MAT-FD to hold (1 tuple over 21), that is: $[7]Patient \overset{4.76}{\underset{10}{\rightarrow}} Pressure$.

Over a relation $\mathbf{r}$, several MAT-FDs can be discovered. However, minimal MAT-FDs are of particular interest, because many other MAT-FDs can be derived from the minimal one. We define a minimal MAT-FD as follows:

Definition 21 (*Minimal* MAT-FD)**.** Given a MAT-FD over $\mathbf{r}$, we define $[k]X \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$ to be minimal for $\mathbf{r}$ if $\mathbf{r} \vDash X \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$ and $\forall X' \subset X$ we have that $r \nvDash [k]X' \overset{\varepsilon}{\underset{\delta}{\rightarrow}} M$.

We recall that MAT-FD must hold, with an error smaller than ε, over the entire dataset. If we *delete* a tuple inside a SW, that tuple will remain *deleted* in *all* the SWs (either following or preceding the current SW), which include that tuple.

It is also important to remind that DT-FD and MAT-FD are *functional dependencies*, and this means that our mining focus is on extracting properties that describe relationships between attributes, and not between their values. So, in our previous example, we found a dependency $Patient \rightarrow Pressure$, regardless the name of the patient nor the values of pressure.

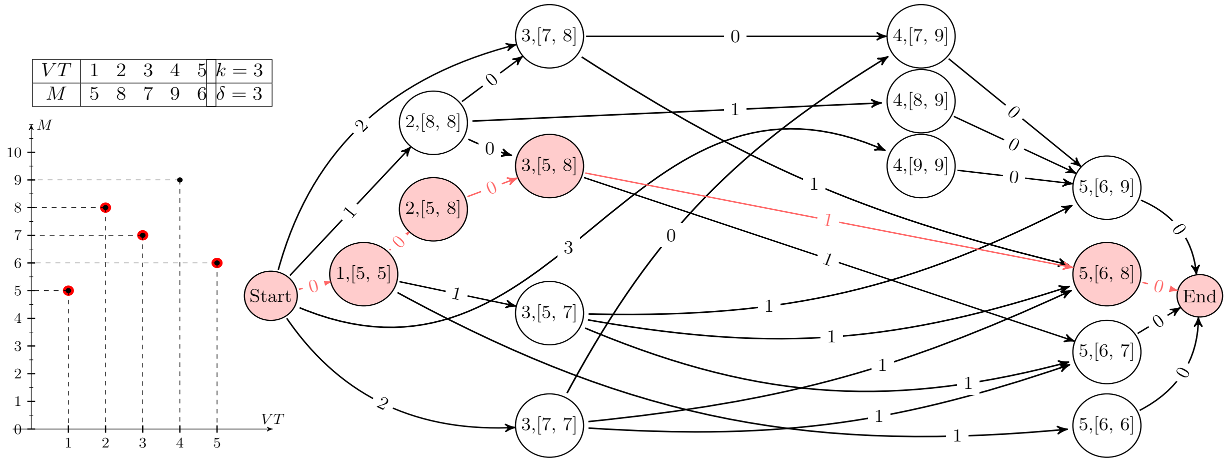


Figure 4-2: An example of graph build over an instance $\mathbf{r}_x$.

As we will show in Section 4.3, *MAT*-FDs may be used to detect variations in trends of the given measure M. Moreover, let us focus on *MAT*-FDs $[k]X \xrightarrow[\delta]{\varepsilon} M$ where value k is the maximum value for which $\mathbf{r} \models [k]X \xrightarrow[\delta]{\varepsilon} M$ (i.e., $\mathbf{r} \not\models [k']X \xrightarrow[\delta]{\varepsilon} M$ for $k' \geq k$). A small (resp. large) value for such k may indicate that the chosen measure M is quickly (resp. slowly) changing over time in the context determined by X. However, since they operate at the attribute level, *MAT*-FDs per se do not provide information neither about the quality (ascending, descending or stable) of the trends nor about the values of such trends.

4.2 Mining *MAT*-FDs

In this section, we introduce the problem of mining *MAT*-FDs on a given instance $\mathbf{r}$ of a temporal schema R. Moreover, we point out two main computational challenges we addressed in the implementation in our prototype. Given a relational schema $R = D \cup \{M\} \cup \{VT\}$, an instance $\mathbf{r}$ of R, a threshold $0 \leq \varepsilon \leq 1$, a real value $\delta \in \mathbb{R}^{\geq 0}$, and a value $k \in \mathbb{N}$ we focus on the problem of returning a set $\mathcal{S}$ of *MAT*-FDs of the kind $[k']X \xrightarrow[\delta]{\varepsilon} M$:

(i) for every *MAT*-FD $[k']X \xrightarrow[\delta]{\varepsilon} M$ in $\mathcal{S}$, we have $k' \geq k$ and $\mathbf{r} \models [k']X \xrightarrow[\delta]{\varepsilon} M$;

(ii) for every *MAT*-FD $[k']X \xrightarrow[\delta]{\varepsilon} M$ in $\mathcal{S}$, *MAT*-FD $[k]X \xrightarrow[\delta]{\varepsilon} M$ is minimal for $\mathbf{r}$;

(iii) for every *MAT*-FD $[k']X \xrightarrow[\delta]{\varepsilon} M$ in $\mathcal{S}$ we have $\mathbf{r} \not\models [k'']X \xrightarrow[\delta]{\varepsilon} M$ if $k'' > k'$ (i.e., the sliding window is maximal).

A maximal set $\mathcal{S}$ that satisfies conditions (i), (ii), and (iii) is called *complete*. Given a temporal schema $R = D \cup \{M\} \cup \{VT\}$, an instance $\mathbf{r}$ of R, a threshold $0 \leq \varepsilon \leq 1$, a real value $\delta \in \mathbb{R}^{\geq 0}$, and a value $k \in \mathbb{N}$ we are interested in finding a complete set $\mathcal{S}$ w.r.t. such parameters. However, in order to solve such problem we have to deal with two main computational problems: (a) what is the minimum size for a complete set $\mathcal{S}$; (b) what is the computational complexity of deciding if $\mathbf{r} \models [k]X \xrightarrow[\delta]{\varepsilon} M$ for a given *MAT*-FD $[k]X \xrightarrow[\delta]{\varepsilon} M$. Let us consider the first problem. Unfortunately, it may be proved, by easily adapting a result of Kivinen et al. [62], that there exist temporal schemas $R = D \cup \{M\} \cup \{VT\}$, instances $\mathbf{r}$ of R, thresholds $0 \leq \varepsilon \leq 1$, real values $\delta \in \mathbb{R}^{\geq 0}$, and values $k \in \mathbb{N}$, for which the the smallest minimal complete set $\mathcal{S}$ w.r.t. such parameters is exponential in the size of D. However, the instance built to prove such a result does not seem to resemble real case scenarios. Moreover, the number of attributes usually does not exceed 50 and does not grow overtime like the number of tuples. Moreover, we can adapt the procedure from [95] to prune the search space for a minimal complete set $\mathcal{S}$. Problem (b) is polynomial in the size of $\mathbf{r}$ but it requires to design a new algorithm to be solved. The rest of this section is devoted to the description of such algorithm. For the sake of brevity, given an instance $\mathbf{r}$ of R,a set $X \subseteq D$, and an element $x \in \pi_X(\mathbf{r})$ we denote with $\mathbf{r}_x$ the instance $\sigma_{X=x}(\mathbf{r})$. It may be easily proved that given an instance $\mathbf{r}$ of R and a *MAT*-FD $[k]X \xrightarrow[\delta]{\varepsilon} M$ we have that $\mathbf{r} \models [k]X \xrightarrow[\delta]{\varepsilon} M$ if and only if:

$$\sum_{x \in \pi_x(\mathbf{r})} \left(\max_{\mathbf{r}' \subseteq \mathbf{r}_x, \mathbf{r}' \models X \xrightarrow[\delta]{} M} |\mathbf{r}'| \right) \geq (1-\varepsilon)|\mathbf{r}|.$$

The above result guarantees the soundness and completeness of the following procedure for verifying whether or not $\mathbf{r} \models [k]X \xrightarrow[\delta]{\varepsilon} M$. Let $\mathbf{r} = \mathbf{r}_{x_1} \cup \ldots \cup \mathbf{r}_{x_n}$, where $\pi_X(\mathbf{r}) = \{x_1, \ldots, x_n\}$ then:

1. initialize $C = 0$
2. for each $1 \leq i \leq n$:
   (a) compute $max_{x_i} = \max_{\mathbf{r}' \subseteq \mathbf{r}_{x_i}, \mathbf{r}' \models [k]X \xrightarrow[\delta]{} M} |\mathbf{r}'|$
   (b) $C = C + max_{x_i}$
3. if $C \geq (1-\varepsilon)|\mathbf{r}|$ then return $\mathbf{r} \models [k]X \xrightarrow[\delta]{\varepsilon} M$ else return $\mathbf{r} \not\models [k]X \xrightarrow[\delta]{\varepsilon} M$.

Now we focus on the only interesting point 2.a of the procedure in which we have to compute

$$max_x = \max_{\mathbf{r}' \subseteq \mathbf{r}_x, \mathbf{r}' \models [k]X \xrightarrow[\delta]{} M} |\mathbf{r}'|.$$

This means that the problem of finding max_x may be encoded as the following integer linear programming problem (for each $t \in \mathbf{r}_x$ let b_t be the boolean variable associated to t):

$$\begin{array}{l} \max(\Sigma_{t \in \mathbf{r}_x} b_t) \\ b_t + b_{t'} \leq 1, \left(\begin{array}{l} \text{(for each } t, t' \in \mathbf{r}_x \text{ with} \\ |t[VT] - t'[VT]| \leq k \\ \text{ and } |t[M] - t'[M]| > \delta \end{array} \right) \\ b_t \in \{0,1\}, \text{ for each } t \in \mathbf{r}_x \end{array} \quad (4.1)$$

Algorithm: $\textsc{MaxInstance}(\mathbf{r}_x, \delta, k)$

```
$N \leftarrow E \leftarrow \emptyset$
$G \leftarrow (N, E, \mathcal{W}_E)$
$VTimes \leftarrow SORT(\pi_{VT}(\mathbf{r}_x))$
for $i \leftarrow 0 \ldots |VTimes| - 1$
    $ct \leftarrow VTimes[i]$
    $Val_{\leq ct} \leftarrow \sigma_{ct-k \leq VT \leq ct}(\mathbf{r}_x))$
    $Val_{=ct} \leftarrow \pi_Y(\sigma_{VT=ct}(\mathbf{r}_x))$
    for each $(lb, ub) \in Val_{\leq ct} \times Val_{\leq ct}$
        if $\left( \begin{array}{c} lb \leq ub \wedge ub - lb \leq \delta \wedge \\ \exists v (v \in Val_{=ct} \wedge lb \leq v \leq ub) \end{array} \right)$
          then $N \leftarrow N \cup \{(ct, [lb, ub])\}$
for each $((ct, [lb, ub]), (\overline{ct}, [\overline{lb}, \overline{ub}])) \in N \times N$
    if $\left( \left\{ t \in \mathbf{r}_x : \begin{array}{c} \overline{ct} - k \leq t[VT] \leq ct \wedge \\ (t[M] < \overline{lb} \vee t[M] > \overline{lb}) \end{array} \right\} = \emptyset \right)$
        $e \leftarrow \{((ct, [lb, ub]), (\overline{ct}, [\overline{lb}, \overline{ub}]))\}$
        $E \leftarrow E \cup \{e\}$
        $count_{\overline{ct}} \leftarrow |\{t \in \mathbf{r}_x : lb \leq t[M] \leq ub\}|$
        $\mathcal{W}(e) \leftarrow \left|\{t \in \mathbf{r}_x : ct < t[VT] \leq \overline{ct}\}\right| - count_{\overline{ct}}$
$N \leftarrow N \cup \{start, end\}$
for each $(ct, [lb, ub]) \in N \setminus \{start, end\}$
    $e \leftarrow (start, (ct, [lb, ub]))$
    $\overline{e} \leftarrow ((ct, [lb, ub]), end)$
    $E \leftarrow \{e, \overline{e}\}$
    $count_{ct} \leftarrow |\{t \in \mathbf{r}_x : lb \leq t[M] \leq ub\}|$
    $\mathcal{W}(e) \leftarrow |\{t \in \mathbf{r}_x : t[VT] \leq ct\}| - count_{ct}$
    $\mathcal{W}(\overline{e}) \leftarrow |\{t \in \mathbf{r}_x : t[VT] > ct\}|$
return $|\mathbf{r}_x| - DAG\_SHP(G, start, end)$
```

Figure 4-3: The pseudocode for finding the maximum cardinality of a set $\mathbf{r}' \subseteq \mathbf{r}_x$ which satisfies $\mathbf{r}' \models [k]X \xrightarrow[\delta]{\epsilon} M$.

However, from the above encoding we can just infer that the sub-problem of finding max_x belong to the complexity class *NP*. Fortunately, we prove that the problem of finding max_x may be solvable in polynomial time. An algorithm solves this problem is shown in Figure 4-3. The idea underlying the algorithm in Figure 4-3 consists of building a weighted directed acyclic graph ($WDAG$) $G = (N, E, \mathcal{W}_E)$ where N is a finite set of nodes, $E \subseteq N \times N$ and $\mathcal{W}_E : E \rightarrow \mathbb{N}^{\geq 0}$. With the exception of two special nodes *start* and *end* each node in N is a triple $(ct, [lb, ub])$ where $ct \in \pi_{VT}(\mathbf{r}_x)$ and $[l, u]$ is an interval (where lb stands for “lower bound” and ub stand for “upper bound”) with $lb, ub \in \pi_M(\mathbf{r}_x)$ and length at most δ $(ub - lb \leq \delta)$.

A triple $(ct, [l, u])$ belongs to N if and only if:

- there exists $t \in \mathbf{r}_x$ such that $t[VT] = ct$ and $lb \leq t[M] \leq ub$ (i.e., there exists at least one tuple at time ct witnessing a value of M included in the interval $[lb, ub]$);
- there exists $t, t' \in \mathbf{r}_x$ with $ct - k \leq t[VT], t'[VT] \leq ct$ with $t[M] = lb$ and $t'[M] = ub$ (i.e., both values lb and ub are witnessed by some tuples t and t' whose VTs are included in the interval $[ct - k, ct]$).

A pair $(ct, [lb, ub]), (ct', [lb', ub']) \in N \times N$ belongs to E if and only if:

- $ct < ct'$;
- $ct < ct' - sw$ or for every tuple $t \in \mathbf{r}_x$ such that $ct - sw \leq t[VT] \leq ct \wedge lb \leq t[M] \leq ub$ we have $lb' \leq t[M] \leq ub$. For every $(ct, [lb, ub]) \in N$ we have $(start, (ct, [lb, ub]))$, $((ct, [lb, ub]), end) \in E$.

Finally, for each $e = ((ct, [lb, ub]), (ct', [lb', ub']))$ in E we have:

$$\mathcal{W}_E(e) = \left| \left\{ t \in \mathbf{r}_x : \begin{array}{c} t[VT] \in (ct, ct') \vee \\ (t[VT] = ct' \wedge t[M] \notin [lb', ub']) \end{array} \right\} \right|.$$

For each $e = (start, (ct, [lb, ub]))$ in E we have:

$$\mathcal{W}_E(e) = \left| \left\{ t \in \mathbf{r}_x : \begin{array}{c} t[VT] < ct \vee \\ (t[VT] = ct \wedge t[M] \notin [lb, ub]) \end{array} \right\} \right|.$$

For each $e = ((ct, [lb, ub]), end)$ in E we have:

$$\mathcal{W}_E(e) = \left| \left\{ t \in \mathbf{r}_x : \ t[VT] > ct \ \right\} \right|.$$

Once we have built the graph G we are interested in finding the value of the shortest path that goes from *start* to *end* in G. As a matter of fact, it is possible to prove with a technique similar to the one used in [29] that the value v of the minimum shortest path between *start* and *end* in G represents the minimum cardinality for a set $\mathbf{r}' \subseteq \mathbf{r}$ such that $\mathbf{r}_x \setminus \mathbf{r}' \models X \to M$ and thus $|\mathbf{r}_x| - v$ is the solution of the linear integer problem 4.1. For the complexity analysis we have that the $|N| \leq |\pi_{VT}(\mathbf{r}_x)| \cdot |\pi_M(\mathbf{r}_x)|^2 \leq |\mathbf{r}_x|^3$ and thus $|E| \leq |\mathbf{r}_x|^6$. The classical shortest path algorithm on DAGs works in $\mathcal{O}(|N| + |E|)$ steps and thus the complexity of our algorithm turns out to be $\mathcal{O}(|\mathbf{r}_x|)$. Let $\pi_X(\mathbf{r}) = \{x_i, \ldots, x_n\}$ when we check the single *MAT*-FD we call the above algorithm for every value $x_i \in$ and thus the complexity of checking a single *MAT*-FD is $\mathcal{O}(|\mathbf{r}_{x_1}|^6) + \ldots + \mathcal{O}(|\mathbf{r}_{x_n}|^6)$ which is bounded by $\mathcal{O}(|\mathbf{r}|^6)$. Figure 4-2 shows an instance of $\mathbf{r}_x$ (the value of X is omitted since all the tuples are supposed to agree on such value) and the corresponding WDAG built using $k = \delta = 3$. In Figure 4-2, nodes belonging to the shortest path from *start* to *end* are highlighted as well as the relative tuples in the graphical representation of $\mathbf{r}_x$. As we detect when we implemented the algorithm in Figure 4-3 an $\mathcal{O}(|\mathbf{r}_{x_n}|^6)$ may affect dramatically the performance of the mining process (e.g., for a relatively small instance of 100 tuples you may need to build a graph with 10^{12} edges). However, there are several optimizations we applied in order to deal with such complexity issue. They do not affect the asymptotic complexity, nevertheless they dramatically improve the performances of the the algorithm on real-world instances. In the following we will describe one of such optimizations along with an example. It immediately follows from the definition of W_E that for every of edge (n, n') in E such that there exists a path $n_1, n_2, \dots n_m$ in G with $n_1 = n$ and $n_m = n'$ we have that $\mathcal{W}_E(n, n') > \sum_{i=1}^{m-1} \mathcal{W}_E(n_i, n_{i+1})$. We call such edges *useless*. It easy to prove that removing useless edges from E does not affect the value for any shortest path in G then we can restrict ourselves to a subset E' of E such that $E' = \{(n, n') \in E : \ (n, n') \text{ is not useless } \}$. Figure 4-2 represents an instance of $\mathbf{r}_x$ and the relative graph G build on $\mathbf{r}_x$ with $\delta = sw = 3$ devoid of useless edges.

4.3 Mining Clinical Data

4.3.1 The Dataset

In order to motivate and validate our approach, we consider the clinical domain of intensive care and in particular we apply our solution to MIMIC (described in details in subsection 2.3.1). Data from Intensive Care Unit (ICU) are particularly useful for testing *DT*-FD and *MAT*-FD, because they contain several measurements of high temporal resolution parameters such as hourly vital signs (heart rate, blood pressures, oxygen saturation, and so on). Moreover, these are the type of data where their measures are strongly affected by daily fluctuation, measurement error, bad reporting and so on, and having a *delta* helps to extract more significant information. Before proceeding with the mining procedure for *MAT*-FDs, an ETL process (Extract, Transform, Load) is required. This process allows us to extract data from MIMIC database and transform them in order to create tables with all the dimensions and a measure we want to analyze. In particular, we focused on table ChartEvents that contains all the charted data available for a patient; ICUstays that defines a single ICU stay; Prescriptions that contains all the prescriptions of a certain drug to a patient; Diagnoses_ICD that contains ICD-9 [43] (International Classification of Diseases, 9th revision) diagnoses for patients; Admissions that defines patients hospital admissions. From these tables, we extract and transform some attributes as follows.

- *VT* - *VT* (Valid Time) represents dates in which patients are admitted, i.e. for each day a patient is admitted, there will be a tuple with the date of that day as VT. *VT* corresponds completely with the date (without the time) from attribute ChartTime (table ChartEvents), that is the date at which a measurement is charted (and in almost all cases, this is the date which best matches the date of actual measurement). Moreover, *VT* represents also dates from table Prescriptions. However, in MIMIC the prescription of a drug to a patient is represented with an interval formed by StartDate and EndDate, so we transformed each interval in a punctual representation (i.e. a tuple with a VT value for each day the prescription is valid for a patient). Because we retrieve VT from multiple tables, we decide to set the granularity to the finest one available on all the tables considered.

- Subject_ID - A unique identifier which specifies an individual patient.
- Hadm_ID - Represents a single patient's admission to the hospital. Let us remember that a patient could experience more than one admission.
- Language - Language of the patient.
- Ethnicity - Ethnicity of the patient.
- ICD9_Code - Contains the code corresponding to the diagnosis assigned to the patient. In particular, we selected the most frequent diagnosis to the patient.
- Admission_Type - describes the type of the admission (elective, urgent, newborn or emergency).
- LOS - Represents the length of stay for the patient for the considered ICU stay, which may include one or more ICU units. In our analysis, we extract the average length of stay for each admission.
- Drug - From table "Prescriptions", we extract Drug, that is the name of the most administered drug in a day to a patient.
- ValueNum - ValueNum is extracted from table ChartEvents. Every measure charted is identified by an ItemID, and filtering by it allows us to create a table with the selected measure as a new attribute. In other words, we create a table with all the attributes previously described, adding a new attribute that represents the measure we filter by ItemID (that is heart rate, blood pressure, and so on). To preserve consistency with other attributes, we extract the daily average value of the measure in order to maintain only a tuple for each day a patient is admitted.

Finally, let us recall that all dates represented in the MIMIC database have been shifted to protect patient confidentiality. MIMIC database contains data collected in 12 years, but these data are randomly distributed over 110 years. However, dates have been maintained internally consistent for a given patient. This distribution nullifies every temporal relation between patients, but to preserve the original time window of 12 years, we decide to delete the offset between all patient's first admission (i.e., after our processing the first day of the first admission for all patients is the same).

After our transformation process, we finally obtain 2 tables for each measure we consider, that are *heart rate*, *respiratory rate*, *SpO_2*, *systolic arterial blood pressure*, *diastolic arterial blood pressure*. The first table represents data of only those patients for which a given measure is charted during the whole hospitalization period. The second table is a subset the first one, and it contains only data for patients whose admissions last at least 7 days. From now on, we refer to the first tables as group TG_1 and to the second ones as group TG_2. Table 4.3 collects some statistics about the patients we selected for our mining.

	TG_1	**TG_2**
Average number of patients	33.864	7.945
Average hospitalization days for a patient	7,59	19,61
Average duration of hospitalization in days	6,15	13,69

Table 4.3: Statistics about patients selected for our mining, and their admission(s).

Measure	ε	δ	Tested *MAT*-FDs		Valid *MAT*-FDs		Total Rows		Processing Time (hh:mm:ss)	
			TG_1	TG_2	TG_1	TG_2	TG_1	TG_2	TG_1	TG_2
Heart rate	0.1	10	32	50	5	6	345.020	233.138	05:26:28	05:27:07
Respiratory rate	0.1	3	39	55	5	9	261.238	157.950	00:08:12	00:04:29
SpO_2	0.1	3	53	52	6	6	261.009	157.677	00:31:34	00:25:31
Arterial BP [Systolic]	0.1	10	39	55	6	9	138.185	83.818	03:41:40	03:36:07
Arterial BP [Diastolic]	0.1	10	47	53	7	6	135.848	82.556	01:15:31	01:11:15

Table 4.4: Summary of *MAT*-FDs mining.

4.3.2 System Configuration

For mining *MAT*-FD on the MIMIC subset depicted in subsection 4.3.1, we developed a running prototype for mining analysis: *SW-MATFDminer* (Sliding Windows Multi Approximate Temporal Functional Dependency Miner). SW-MATFDminer is a Java based system that extract rules of multi approximate temporal functional dependency for sliding window (SW) temporal grouping. The prototype allows the user to configure multiple parameters, such as: delta and epsilon approximations, minimum and maximum *SWs* accepted, name of attributes to use as measures. Moreover, there are flags to enable multi-threaded mining and resuming function. SW-MATFDminer was tested on a machine with a Intel® Core™ i7-6700 and 32 GB of RAM, equipped with Windows 10 64-bit, Java 1.8, and PostgreSQL 9.6.

4.3.3 Results

For our test, we set SW-MATFDminer parameters as follows: the maximum percentage of tuples to be deleted (ε) is 0.1; delta (δ) has a value of 10 for heart rate, systolic and diastolic arterial blood pressure and 3 for respiratory rate and SpO_2; minimum window size is 1 day; maximum window size is 12 years, that is the maximum distance in time between the tuples we consider. This means that the mining algorithm returns the max. window size for *MAT*-FD to hold within this specified interval. If the resulting SW equals the maximum one, then the *MAT*-FD is considered atemporal.

Table 4.4 summarizes the results of our mining. Overall, we analyze 10 tables, that is a table of TG_1 and TG_2 for all the five measures considered. Each table consists of 8 attributes and a mining algorithm would need 2560 (i.e. $10 \cdot 2^8$) functional dependencies to validate. However, due to pruning operations, SW-MATFDminer tested only 475 dependencies in

Group	MAT-**FD**
TG_1	$[1]Hadm\_ID \xrightarrow[10]{0.1} Heart\ rate$
	$[1]LOS, Drug, ICD9\_Code \xrightarrow[10]{0.1} Heart\ rate$
	$[1]Subject\_ID \xrightarrow[3]{0.1} Resp.\ rate$
	$[1]LOS, Drug, ICD9\_Code \xrightarrow[3]{0.1} Resp.\ rate$
	$[12]Subject\_ID \xrightarrow[3]{0.1} SpO_2$
	$[13]Hadm\_ID \xrightarrow[3]{0.1} SpO_2$
	$[1]Hadm\_ID \xrightarrow[10]{0.1} Systolic\ arterial\ BP$
	$[1]LOS, Drug, ICD9\_Code \xrightarrow[10]{0.1} Systolic\ arterial\ BP$
	$[2]Hadm\_ID \xrightarrow[10]{0.1} Diastolic\ arterial\ BP$
	$[4]LOS, Drug, ICD9\_Code \xrightarrow[10]{0.1} Diastolic\ arterial\ BP$
TG_2	$[2]LOS, Drug, ICD9\_Code \xrightarrow[10]{0.1} Heart\ rate$
	$[1]LOS, Language, ICD9\_Code \xrightarrow[10]{0.1} Heart\ rate$
	$[1]Subject\_ID \xrightarrow[3]{0.1} Resp.\ rate$
	$[2]LOS, Drug, ICD9\_Code \xrightarrow[3]{0.1} Resp.\ rate$
	$[7]Subject\_ID \xrightarrow[3]{0.1} SpO_2$
	$[7]Hadm\_ID \xrightarrow[3]{0.1} SpO_2$
	$[1]Subject\_ID \xrightarrow[10]{0.1} Systolic\ arterial\ BP$
	$[1]LOS, Drug, ICD9\_Code \xrightarrow[10]{0.1} Systolic\ arterial\ BP$
	$[2]Hadm\_ID \xrightarrow[10]{0.1} Diastolic\ arterial\ BP$
	$[2]Subject\_ID \xrightarrow[10]{0.1} Diastolic\ arterial\ BP$

Table 4.5: An excerpts of our results with 2 valid MAT-FDs for each mined table. The length of the maximum sliding window found, is expressed in days.

21 hours 47 minutes and 54 seconds, and 65 of them are valid. The processing time for a measure, as it can be seen in table 4.4, may be highly variable. In fact, it depends by the number of tuples, the amount of MAT-FDs to tested, but it is also influenced by the number of nodes and edges we had to create. SW-MATFDminer implements a dichotomic search algorithm to extract the maximum sliding windows for which the MAT-FD holds. The length of a sliding window determines how many nodes and edges have to be created, and this affects the processing time. Table 4.5 shows an excerpt of our results and, in the following,

we describe some meaningful MAT-FDs that have been discovered by SW-MATFDminer:

- $[7]Hadm\_ID \xrightarrow[3]{0.1} SpO_2$: this dependency points out that SpO_2 varies slowly during patient's hospitalization. As depicted in table 4.3, the average duration of hospitalization is 13.69 days for patients in TG_2, and a maximum sliding window of 7 days indicates that the level of oxygen saturation is fairly stable. However, it is not constant during the whole hospitalization time-span, otherwise the maximum sliding window would have been larger.

- $[4]LOS,Drug,ICD9\_Code \xrightarrow[10]{0.1} Diastolic\ arterial\ BP$: this dependency links patients with the same diagnosis, length of stay in ICU and under the same treatment, with their diastolic arterial blood pressure, in a window of 4 days. As it can be seen in table 4.5, the same dependency holds also for the systolic arterial blood pressure ($[1]LOS, Drug, ICD9\_Code \xrightarrow[10]{0.1} Systolic\ arterial\ BP$). However, it holds with a smaller sliding window (1 day) and this indicates that systolic pressure suffers from an higher fluctuation during the hospitalization.

4.4 Conclusions

In this chapter, we introduced and discussed multi approximate temporal functional dependencies (MAT-FDs). We have evaluated and discussed their expressiveness by means of some examples taken from the clinical domain. More precisely, we proposed an algorithm to mine MAT-FDs over a given data set and we implemented it into a working prototype called SW-MATFDminer. We used SW-MATFDminer to mine data belonging to the clinical domain of intensive care and in particular we applied our solution to the MIMIC database. Our study proves SW-MATFDminer to be an interesting tool for mining clinical data. The derived dependencies may provide physicians with new kind of knowledge underlying medical data.

As future work, we plan to further validate mined MAT-FDs with clinical experts, to focus on the more clinically relevant derived knowledge. Moreover, it would be interesting to extract MAT-FDs from different clinical domains. Finally, a tuning of the proposed prototype will be considered. In particular, SW-MATFDminer could benefit from parallelization techniques applied to distributed computing to reduce processing time.

Chapter 5

Discovering and Analyzing Trend-Event Patterns on Clinical Data

In the previous chapter, we mined MIMIC III EHR dataset to derive a new kind of Approximate Temporal Functional Dependency that focuses on *quantitative* attributes. In this chapter, we use the same ICU dataset to infer a new kind of *temporal pattern* that also focuses on quantitative attributes.

Considering the current literature and the need for significative synthesis information for several medical domains, we propose a new kind of temporal pattern called *Trend-Event Patterns*, namely *TE*-Ps. The *TE*-P family of temporal patterns focus on the interaction of trends and events. For us, a trend is formed by consecutive values of a given measurement attribute that are stationary, increasing, or decreasing under the constraint that all the values of such trend stay within a defined range. In other words, all the values that form a trend are allowed to have negligible variations within some specified threshold. For instance, *TE*-Ps could express concepts like *"patient's systolic blood pressure was rising before the administration of lisinopril then, after the administration, it stabilized"*.

Another contribution in this chapter is the development of a tool, called TEPminer, which implements the algorithm to extract *TE*-Ps. Even though TEPminer has been designed to be a general-purpose tool (i.e., it is able to mine *TE*-Ps regardless of the application domain), we tested it using data coming from intensive care units (ICUs) contained in MIMIC III EHR dataset (that is illustrated in Section 2.3.1). To speed up the mining process, TEPminer exploits multithread functions to analyze data of each patient independently. Finally, we took advantage of OLAP analysis to present some multidimensional analysis on *TE*-Ps, where patterns are evaluated in an aggregate form based on the level of detail we want to use.

The novelties in this work have been published in [75].

5.1 Trend-Event Patterns

In this section, we introduce a new kind of temporal pattern named *Trend-Event Patterns*, *TE*-Ps for short.

Let us recall that with *TE*-Ps we want to express concepts like "A patient's body temperature was *increasing* before the administration of *paracetamol*. After such administration, body temperature was *steady*". In other words, a *TE*-P is a pattern formed by an event E and two different trends for the same measure and dimensions: one before E and one after E. We call those trends $trend^{pre}$ and $trend^{post}$ respectively.

We use Figure 5-1 as a reference to explain *TE*-Ps. In this figure we recreated a typical scenario of raw data for a given measure and for a patient. Our first goal is to discover every possible event, then to eventually derive a trend before and a trend after it. In Figure 5-1 raw measure data are depicted as scattered squares, where every square represents a tuple and its position is given by its *measure* and its valid time VT, i.e. the time when the tuple is valid in the represented real world [103]. Event E is represented by a vertical line, placed on its corresponding VT.

To derive a *TE*-P from this scenario, we need to start from the event, and more specifically from the time when such event happened. From this point we can partition measures, according to their VT. Every tuple before the event could potentially be part of $trend^{pre}$, while every tuple after the event could be in $trend^{post}$. In both trends the first tuple is denoted as t_{start}, while the last one is t_{end}. The tuple in $trend^{pre}$ that is closer to the event is called t^{pre}_{end}, because it is the last tuple of it; while the tuple in $trend^{post}$ that is closer to the event is called t^{post}_{start}, as it represents the beginning of such trend. In our proposal, there can only be one single event for every VT. Thus, if there are many events that occur at the same time, they will be merged together in a single event. For example, let the event be a single drug administered to a patient: if in a certain moment the patient receives two different drugs, the event is formed by the conjunction of these two drugs and it is different from the events formed by these two drugs taken individually.

For us, a trend is graphically depicted as a segment line from t_{start} to t_{end} and, ideally, every other tuple of such trend should lay in this segment. However, this is an unlikely scenario with real data. Thus it is necessary to allow a threshold (called Δ_y) on the segment line, and every tuple of a trend must be within such threshold. Tuple t_{ext_1} represents the violation of the above constraint: a trend from t^{pre}_{start} to t_{ext_1} excludes many tuples in-between and consequently it is not a valid trend.

There are few more constraints necessary to determine a valid trend: some of them are tuple-related and some trend-related.

The time distance between two subsequent tuples must be less than a predefined parameter, called $\max \Delta_{VT}$. Tuple t_{ext_2} is an example of violation of the constraint above: the time distance between that tuple and its predecessor, that is t^{post}_{end}, is greater that $\max \Delta_{VT}$ and consequently it could not be admitted in the trend. Moreover, the number of tuples needed to form a trend must be greater or equal the *minNumTuples* parameter.

As for trend-related constraints, we must assure that a trend has a minimum/maximum duration ($\min \Delta_{duration}/\max \Delta_{duration}$ respectively) and that it begins/ends not too far from

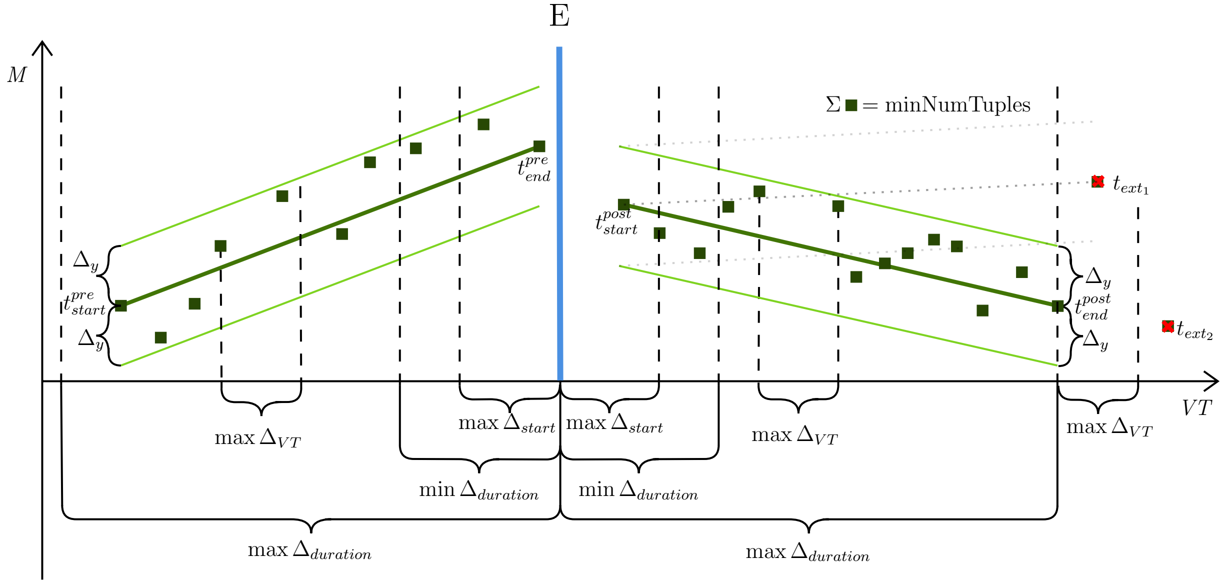


Figure 5-1: This figure shows an example of *TE*-P. We have two trends, one from t^{pre}_{start} to t^{pre}_{end} that precedes event E, and a second one from t^{post}_{start} to t^{post}_{end} right after E. These are valid trends because they respect every constraint. In fact, t_{ext_1} and t_{ext_2}, are external to these trends because they violate Δ_y or $\max \Delta_{VT}$.

the event ($\max \Delta_{start}$). All these constraints will be better described later on this chapter. It is important not to perceive all these constraints as a limitation on the number of retrieved trends. Instead, these parameters are extremely useful for extracting just those trends that clinicians might think are appropriate for the kind of event and measure they are analyzing. Alternatively, they could set these parameter in the opposite way, to mine and to analyze *TE*-Ps coming from an "unexpected" scenario.

As shown in Figure 5-2, every trend that has been retrieved is then labeled as {*increasing, steady, decreasing*} according to its *weighted rate of change* and a predefined threshold called *max increase*.

Now we are ready to give the definition of *TE*-P. A *TE*-P is a pattern with an expression of the form:

$$[trend^{pre};E;trend^{post}]$$

For example, let E be the *"administered drugs"* attribute, and *paracetamol* an $e \in E$. Concepts like "A patient's body temperature was *increasing* before the administration of *paracetamol*. After such administration, body temperature was *steady*" would be written as *TE*-P as:

$$[increasing;paracetamol;steady]$$

From now on, we'll describe all the previous concepts in a more formal way.

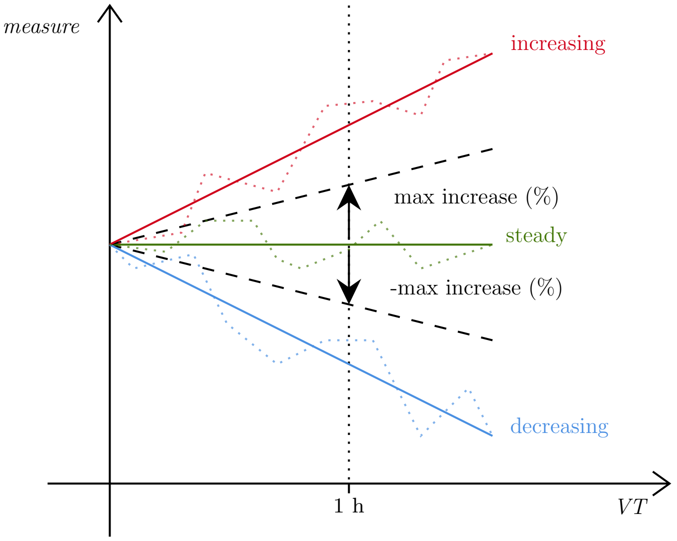


Figure 5-2: In this figure we show how trends are labeled, based on their weighted rate of change.

Let us start with some basic notions about trends. We consider classical temporal schema $R = U \cup \{VT\}$ where R is a set of *atemporal attributes* and VT is an attribute denoting the *valid time* of each tuple. We may safely assume $Dom(VT) = \mathbb{R}$. Given an attribute $M \in U$ we say that M is a *measure* if and only if $Dom(M)$ is a metric space. We assume that for all measures M it holds $Dom(M) = \mathbb{R}$. Now, we give the criteria under which two tuples are connected in order to form a valid trend on some measurement M in U. First, tuples must share the same values for some non-empty *dimensions* $D \subseteq U$. Also, tuples must be temporally ordered using their VT: this means that, given an instance $\mathbf{r}$ of R, two tuples t, t' satisfy $t \rightarrow t'$ if and only if $t[D] = t'[D]$, $t[VT] < t'[VT]$, there is no tuple t'' in $\mathbf{r}$ with $t''[D] = t[D]$ s.t. $t[VT] < t''[VT] < t'[VT]$. In the following, we assume that there are no tuples with the same VT.

Given a trend *tr*, we denote with t_{start} the first tuple of *tr*, and with t_{end} the last one. A trend *tr* is a non-empty set of tuples $tr = \{t_{start}, \ldots, t_{end}\}$ satisfying $t_{start} \rightarrow \ldots \rightarrow t_{end}$. Implicitly, it means that $\forall t_i \in tr$, $t_{start}[VT] < t_i[VT] < t_{end}[VT]$.

As noted before, with *TE*-Ps we are interested in relations between an event, $trend^{pre}$, and $trend^{post}$. Let t_{event} be the tuple that denotes the event. Tuples in $trend^{pre}$ are defined as follows: $\forall t_i \in tr^{pre}$, $t_i[VT] \leq t_{event}[VT]$; while tuples in $trend^{post}$ are defined as $\forall t_i \in tr^{post}$, $t_i[VT] > t_{event}[VT]$.

A *maximal trend tr* is a trend such that for every $t \in \mathbf{r} \setminus tr$ we have that $tr \cup \{t\}$ is no longer a trend. By definition, all the tuples $t_i \in tr$ share the same values for the attributes in D (resp. E), then we denote them with $tr[D]$ (resp. $tr[E]$). Any trend must satisfy all the following condition on measures and on VT to hold (as also shown in Figure 5-1):

- *minNumTuples* - This is a constraint that denote the minimum number of tuples that are needed to form a trend. With just two tuples we can construct a line segment that can form a trend, however this parameter is needed to better define our trend. In other words, there is a trend if:

$$\sum_{k=t_{start}}^{t_{end}} t_k \geq minNumTuples$$

- $\max \Delta_{VT}$ - This parameter indicates the maximum time distance between two adjacent tuples to be part of the same trend. Given two tuples $t_m \in tr$ and $t_{m+1}, t_{m+1} \in tr$ if $t[VT]_{m+1} - t[VT]_m \leq \max \Delta_{VT}$. This is useful to avoid those measures that are taken too far away one another. Let us imagine a scenario where we want to monitor heart rate right after a panic attack: if a measurement is taken 10 hours after the last one, then is no longer meaningful for our trend.

- $\max \Delta_{start}$ - Another condition on VT is given by $\max \Delta_{start}$. For us, a trend is insightful if the measure that is closer to the event, is measured within a negligible time distance, that is $\max \Delta_{start}$. In other words, $\max \Delta_{start}$ represents the maximum distance between a trend and an event E.

- $\min \Delta_{duration}$ - Control the time distance of a trend from an event is of fundamental importance. For this reason we introduce $\min \Delta_{duration}$, that represents the minimum distance of a trend from an event. This parameter is necessary to avoid those trends where most measures of a trend are concentrated in a small time frame that is, also, really close to the event. An example could be the monitoring of blood pressure after the administration of a drug for hypertension: it is not interesting nor meaningful to have measurements of blood pressure every 5 minutes after the administration of a drug to test the long-term effectiveness of such drug.

- $\max \Delta_{duration}$ - The complementary parameter of $\min \Delta_{duration}$ is $\max \Delta_{duration}$. This parameter, that defines the maximum time distance of a trend from an event, is particularly useful when the goal is to extract trends for testing short-term effects of a certain event. As mentioned above, a panic attack is a meaningful example of such situation. In fact, it is important to monitor heart rate right after that event occurs, but it is not so interesting to track its behavior after many hours. However, $\max \Delta_{duration}$ is different from $\max \Delta_{VT}$ because its related to the whole trend, while $\max \Delta_{VT}$ considers couple of adjacent tuples within the trend.

- Δ_y - A trend tr from t_{start} to t_{end} is graphically represented by a *line segment*. Ideally, all tuples in tr must lay in that segment, but it is a highly unlikely situation with real word data. However, we could tolerate a small threshold where the difference between the measure and its ideal position in the segment is negligible. Δ_y represents that threshold, and must hold for all the tuples in *tr*. Given $t_n \in tr$, let $t_i[M]$ the projection of $t_n[M]$ on *tr*, namely:

$$t_i[M] = (t_i[VT] - t_{start}[VT]) * m + t_{start}[M] \tag{5.1}$$

where

$$m = \frac{t_{end}[M] - t_{start}[M]}{t_{end}[VT] - t_{start}[VT]}$$

there is a trend *tr* if $\forall t_n[M] \in \{t[M]_{start}, \ldots, t[M]_{end}\}$ then $t_i[M] - \Delta_y \leq t_n[M] \leq t_i[M] + \Delta_y$.

If all the constraints depicted above hold, then we have a valid trend *tr*. The trend is then labeled using the alphabet $\Sigma = \{$*increasing, steady, decreasing*$\}$. To correctly label a trend, we need to calculate its weighted rate of change (in percentage), that is:

$$changeRate = \frac{t_{end}[M] - t_{start}[M]}{t_{end}[VT] - t_{start}[VT]} * \frac{100}{t_{start}[M]}$$

As shown in Figure 5-2, we define a new parameter *max increase* as a threshold for *changeRate*, and we determine the label of a trend as follows:

- *increasing*: *changeRate* > *max increase*
- *decreasing*: *changeRate* < *-(max increase)*
- *steady*: otherwise

5.2 Mining *TE*-Ps

In this section we describe the algorithm that we used to mine *TE*-Ps. Let Table 5.1 represent a small excerpt of our data. First of all, we need to pick a dimension $d \in D$ that we want to use to group data. In Table 5.1, $D = \{$*Subject_ID, Hadm_ID, ICUstay_ID*$\}$, that represent an unique identifier for a patient, for his/her admission, and for his/her stay in ICU respectively. We partition Table 5.1 in order to group all the tuples of every different *d*, so the algorithm will extract *TE*-Ps only related to *d*.

For each partition *d*, 2 different lists are created:

- measuresList - containing only the measures and *VT* of such partition, ordered by their *VT*
- eventsList - that contains all the events of such partition, their *VT* and the index of the closest measures that preceded them

This separation is useful to speed-up the mining process, because we avoid to check sequentially if every following tuple represents an event or a measure.

In the following, we describe step-by-step the procedure to validate the trend that preceded an event. The validation of a trend post-event is specular to this one, and thus it is omitted. It is fundamental to note that the following algorithm is performed **for every event**. If there are no valid trends for an event, the current event is ignored and the whole process restart for the next one. Figure 5-1 could help to better understand the following steps.

1. Check for *minNumTuples*. Given the current event, we retrieve from eventsList the index (in measuresList) of the closest measure that preceded it. We check if there are a minimum number of tuples (*minNumTuples*) before that measure. Consequently it is also immediate to retrieve the measure right after the event, as it is the next element in measuresList. Then, we check if there are *minNumTuples* from that measure to the end of measuresList. If it doesn't exist, the event is ignored.

2. Check for $\max \Delta_{start}$. Given the considered event and its closest measures that preceded it, we have to check the difference of their VT, that have to be lower or equal than $\max \Delta_{start}$. Otherwise, the event is ignored. From now on, all the constraints to define one extremity of the trend are satisfied, and we call t^{pre}_{end} the closest measure that preceded the event.

3. Discover t^{pre}_{start}. In this step, we initiate a loop through measuresList to find the initial point of the trend, called t^{pre}_{start}. Initially, t^{pre}_{start} is the measure right before t^{pre}_{end}, and for each iteration t^{pre}_{start} becomes the preceding tuple in measuresList if all the measures from t^{pre}_{start} to t^{pre}_{end} satisfy all the constraints. Therefore, the number of iterations corresponds to the number of tuples in the trend. In the following, we explain all the steps performed within a single loop:

   (a) Check for $\max \Delta_{VT}$. It is necessary to check if the temporal distance between two adjacent tuples is less than $\max \Delta_{VT}$. To achieve that, we check the VT difference between t^{pre}_{start} and its subsequent tuple. If this constraint is not satisfied, we exit the loop.

   (b) Check for $\max \Delta_{duration}$. Analogously, it is necessary to check if he temporal distance between t^{pre}_{start} and the event is less than $\max \Delta_{duration}$. Again, if this constraint is not satisfied, we exit the loop.

   (c) Check for Δ_y. As already mentioned in Section 5.1, all the measures of a trend must lay on the segment from t^{pre}_{start} to t^{pre}_{end}, plus a negligible threshold that is Δ_y. Therefore, we have to check that every tuple t_i satisfies this constraint, and to do so we use the equation 5.1 listed in Section 5.1. If even a single measure exceeds Δ_y we cannot have a valid trend with the current t^{pre}_{start} and we exit both loops.

4. Check *minNumTuples*. After t_{start}^{pre} is finally retrieved, we check the number of measures in the current trend, that must be greater than *minNumTuples*, otherwise the current event is ignored.

5. Check $\min \Delta_{duration}$. The retrieval of t_{start}^{pre} is also necessary to check if the trend violates $\min \Delta_{duration}$. In fact, we have to check whether the time distance between t_{start}^{pre} and the event is greater than $\min \Delta_{duration}$ or not. In the first scenario the event is ignored, otherwise we finally have a valid trend.

We mined the trend that preceded an event, and we follow the same steps (but moving from t_{start}^{post} to t_{end}^{post}) to retrieve the trend post-event. If they are both valid trends, they need to be labeled as {increasing, decreasing, steady} based on their weighted rate of change, as shown in Section 5.1. Finally, we need to check whether an event influenced them or not. To verify this condition, we create a new segment from t_{start}^{pre} to t_{end}^{post}: if all the tuples in-between are included in Δ_y, then there is only a trend from t_{start}^{pre} to t_{end}^{post}. Thus, the event is not influencing the considered measure. After all these processes, we finally obtain a valid ***TE*-P**.

5.2.1 Optimized Algorithm for Mining *TE*-Ps

The algorithm shown above is what we proposed in our paper [75]. Since then, we thought on a way to improve it and to reduce its complexity. In particular, the admissibility of a trend given a new tuple is particularly time consuming, because it is necessary to re-check if every tuple within the new trend satisfies the constraint of Δ_y, as shown in the part 3c of the algorithm. A possible solution could be inherited from the method called *Scan Along Polygonal Approximation (SAPA-2)* proposed in [8] and its extension, called *Sample skipping method with amplitude and time error limitation*, proposed in [17].

The basic idea of this algorithm is to analyze and store only the minimum highest and the maximum lower slopes given by the trends of the previous measures. With this approach, testing the admissibility of a tuple in the trend requires only a comparison between its slope and these two bounds.

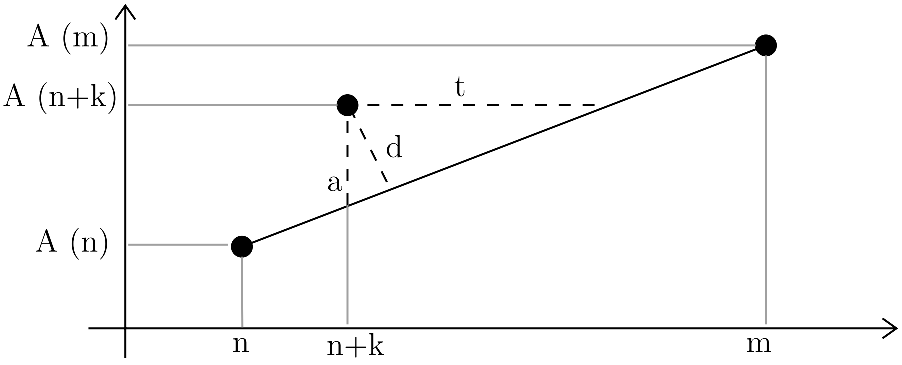


Figure 5-3: Definition of distance, amplitude and time errors.

As shown in Figure 5-3, let us consider three tuples at instants n, $n + k$ and m with n < n + k < m, and having amplitudes A(n), A(n + k) and A(m), respectively. Let us consider the distance d between the real measure at instant $n + k$ and the segment joining the tuples at instants n and m: this technique aims at limiting the distance error ($d < d_{admissible}$). However, the distance error contains a square root and the calculation could be excessively long for the amount of tuples we have to analyze. Thus it is preferred to use the a/t relationship: $\frac{|a|}{|t|} = \frac{A(m) - A(n)}{m - n}$. By simplifying the distance error test, we can impose: $a < d_{admissible}$ or $t < d_{admissible}$, according to the slope of the segment that interpolates two tuples. In this way, as long as the slope of the signal is smaller than 1 only an amplitude error test is run. When the absolute value of the slope exceeds 1, the test is run on t, i.e. on the time error. The amplitude and time tests are run when the slope of the interpolating segments is smaller or greater than a specified threshold.

5.2.2 Multidimensional Modeling of Trends for OLAP Analysis

After all the *TE*-Ps have been retrieved, we represent them as a data-cube to analyze them in some aggregate form. The data cube structure offers a powerful data retrieval and data analysis environment, in addition to analytical modeling capabilities. In our cube, we used as *measures* all the starting/ending points for the trends, as well as the number of rows, slopes, and duration of the trends. All the parameters listed in Table 5.2 and Table 5.3, the event name, and patient ID are some of the *dimensions* we used to analyze our *TE*-Ps. As shown in Figure 5-4, a better representation of *dimensions* and *measures* used in our OLAP analysis is given by the *Dimensional Fact Model* (DFM), that is a graphical formalism specifically designed to support the conceptual modeling phase in a datawarehouse project. Let us denote that, in this aggregate analysis, we could group only those events that are equal. For example, if we use drug administration attribute as event, the administration of paracetamol and its influence on vital signs is analyzed separately from paracetamol administered in conjunction with codeine.

5.3 Mining Clinical Data

5.3.1 The Dataset

We applied our techniques in the clinical domain of Intensive Care Unit (ICU) present in MIMIC (described in details in subsection 2.3.1) in order to motivate and validate our algorithm.

ICU data contain multiple measurements of high temporal resolution parameters such as hourly vital signs (heart rate, blood pressures, oxygen saturation, body temperature and so on) and these measures could potentially form a trend. Moreover they record every drug administered to a patient, and this is what we select as an event.

All the data we are using from MIMIC need a process of E.T.L. (Extract, Transform, Load) before the mining procedure could start. We focused on 3 different tables: ChartEvents,

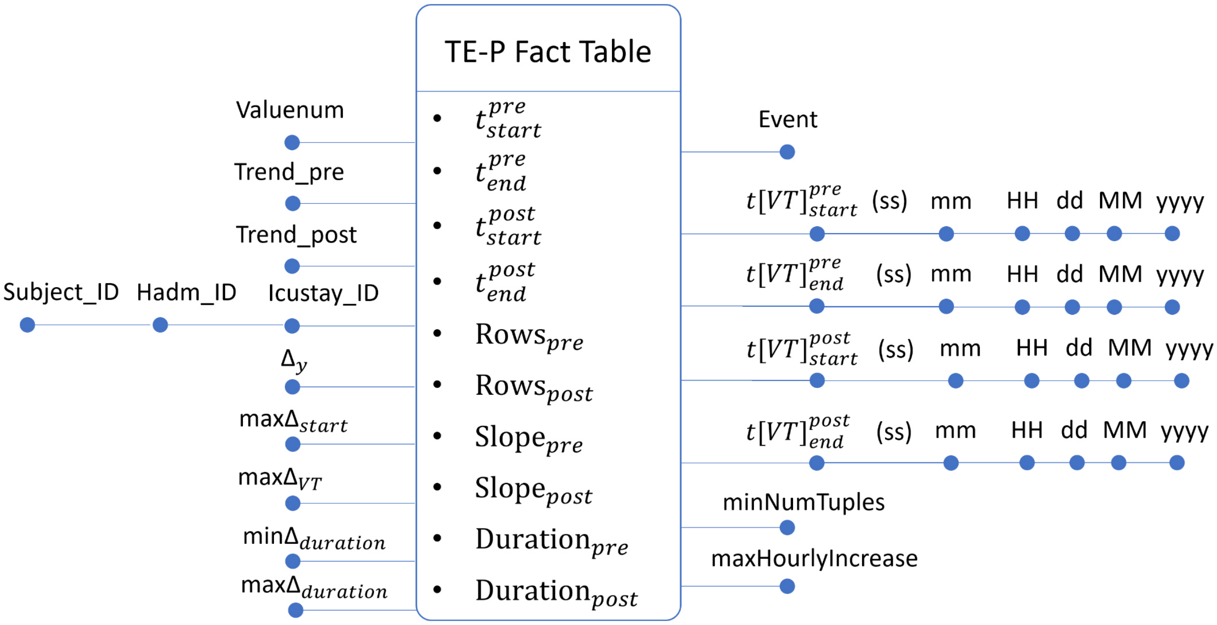


Figure 5-4: Dimensional Fact Model (*DFM*) used for our OLAP analysis.

where all the vital signs of every patient are stored; InputEvents_MV, that contains any fluid administered to the patients, such as oral or tube feedings or intravenous solutions containing medications; D_Items, that is the definition table, where we can correctly label every event related to a patient.

From these tables, we extract and transform some attributes as follows:

- Subject_ID - A unique identifier that specifies an individual patient.
- Hadm_ID - Represents a single patient's admission to the hospital. Let us remember that a patient could experience more than one admission.
- ICUstay_ID - A unique identifier that defines any single ICU stay.
- VT - Represents the *Valid Time (VT)*, that is the date and time in which a vital sign is charted. In almost all cases, this is the date which best matches the date of actual measurement. *VT* is also used to identify the moment when a fluid have been administered to a patient from InputEvents_MV table.
- Event - From the *label* attribute in D_Items, we create Event to identify every drug admitted to patients during their stay. If multiple drugs are admitted to a patient at the same time (i.e., same *VT*), they are joined together and not treated as separate events. For example, if a patient received paracetamol and epinephrine in the same *VT*, their influence on vital signs is analyzed together even if the medications are stored separately.

- VALUENUM - VALUENUM is extracted from table CHARTEVENTS. Every measure charted is identified by an ITEMID, and the filtering process by this attribute is used for creating a new table with the selected measure as a new attribute.

  In other words, for each attribute that represents the measure we filter by ITEMID (e.g., blood pressure, respiratory rate, heart rate, and so on) we create a new table with it and all the other attributes previously described.

At the end of our transformation process, we obtain a table for each measure we consider, that are *heart rate*, *respiratory rate*, SpO_2, *systolic arterial blood pressure*, *diastolic arterial blood pressure*, and *body temperature*.

Table 5.1 represents an excerpt of the table that we create, where D={SUBJECT_ID, HADM_ID, ICUSTAY_ID}, M=VALUENUM (Heart rate), VT=VT and E=EVENT. Attribute ROW # is just added in the table to facilitate the explanation.

In this table we have data from 2 different patients, one with SUBJECT_ID 33 864 and the other one with 23 460. Patient 33 864 had 2 different stays in ICU during the same admission, one with ID 1515, and the other one with 1523. This is useful to explain the different outcome based on the attribute of grouping: if we group using SUBJECT_ID or HADM_ID, we should consider all the measures from row 1 to 22, to check if they form trends for events *atropine* and *paracetamol*. On the contrary, if we group using ICUSTAY_ID we need to separate the measures from row 1 to 11, to the ones from row 12 to 22. Finally, let us focus on the administration of *atropine* in rows 6 and 28. Even though they represent the same drug admitted in the same VT, they are treated as two separate events because they are referring to different patients.

5.3.2 System Configuration

To mine *TE*-Ps on the MIMIC subset depicted in subsection 5.3.1, we developed a running prototype for mining analysis: *TEPminer* (Trend-Event Pattern miner). TEPminer is a Java based system that extract trend-event patterns, and allows the user to configure multiple parameters, such as: Δ_y, $\max \Delta_{start}$, $\max \Delta_{VT}$, $\min \Delta_{duration}$, $\max \Delta_{duration}$, *minNumTuples*, max increase (defined as max hourly increase), and the name of attributes to use as: ID for the patient, *VT*, trends and events. Moreover, because of the grouping by *dimension*, the extraction of *TE*-Ps is bound to single patients. Consequently, TEPminer exploits multithread functions to speed-up the mining process, analyzing data of each patient independently. TEPminer was tested on a machine with a Intel® Core™ i7-6700 and 32 GB of RAM, equipped with Windows 10 64-bit, Java 10, and PostgreSQL 9.6.2.

TEPminer is also designed to store any possible meaningful parameter related to trends and events to allow us to analyze *TE*-Ps from many different points of view. In fact, after the mining process, we used Saiku[1] as an OLAP software to perform multidimensional analysis on *TE*-Ps.

[1] https://www.meteorite.bi/products/saiku

Row #	**VT**	**Subject_ID**	**Hadm_ID**	**ICUstay_ID**	**Event**	**ValueNum**
1	07 Jan 2011 11:25	33 864	7945	1515	-	55
2	07 Jan 2011 11:55	33 864	7945	1515	-	57
3	07 Jan 2011 12:25	33 864	7945	1515	-	58
4	07 Jan 2011 12:55	33 864	7945	1515	-	62
5	07 Jan 2011 13:25	33 864	7945	1515	-	65
6	07 Jan 2011 13:30	33 864	7945	1515	Atropine	-
7	07 Jan 2011 13:55	33 864	7945	1515	-	65
8	07 Jan 2011 14:25	33 864	7945	1515	-	62
9	07 Jan 2011 14:55	33 864	7945	1515	-	67
10	07 Jan 2011 15:25	33 864	7945	1515	-	69
11	07 Jan 2011 15:55	33 864	7945	1515	-	62
12	08 Jan 2011 14:25	33 864	7945	1523	-	55
13	08 Jan 2011 14:55	33 864	7945	1523	-	57
14	08 Jan 2011 15:25	33 864	7945	1523	-	58
15	08 Jan 2011 15:55	33 864	7945	1523	-	62
16	08 Jan 2011 16:25	33 864	7945	1523	-	60
17	08 Jan 2011 16:55	33 864	7945	1523	-	61
18	08 Jan 2011 17:25	33 864	7945	1523	-	62
19	08 Jan 2011 17:30	33 864	7945	1523	Paracetamol	-
20	08 Jan 2011 17:55	33 864	7945	1523	-	67
21	08 Jan 2011 18:25	33 864	7945	1523	-	69
22	08 Jan 2011 18:55	33 864	7945	1523	-	71
23	07 Jan 2011 11:25	23 460	6941	1569	-	45
24	07 Jan 2011 11:55	23 460	6941	1569	-	46
25	07 Jan 2011 12:25	23 460	6941	1569	-	48
26	07 Jan 2011 12:55	23 460	6941	1569	-	43
27	07 Jan 2011 13:25	23 460	6941	1569	-	44
28	07 Jan 2011 13:30	23 460	6941	1569	Atropine	-
29	07 Jan 2011 13:55	23 460	6941	1569	-	55
30	07 Jan 2011 14:25	23 460	6941	1569	-	52
31	07 Jan 2011 14:55	23 460	6941	1569	-	59
32	07 Jan 2011 15:25	23 460	6941	1515	-	62
33	07 Jan 2011 15:55	23 460	6941	1515	-	64

Table 5.1: Excerpt of the table we create to extract *TE*-Ps from *heart rate*. Here we can observe different measurement of heart rate, for two different patients during three ICU stays, interrupted by the administration of *atropine* or *paracetamol*.

$\max \Delta_{start}$	$\max \Delta_{VT}$	$\min \Delta_{duration}$	$\max \Delta_{duration}$	$\Delta_y(\%)$	$\min numTuples$
{1;2} h	6 h	2 h	{8;12} h	{0.5;1;2.5;5;10;20}	10

Table 5.2: Running parameters for TEPminer.

	HR	**DBP**	**SBP**	**RR**	**Temp**	**SpO$_2$**
Max Hourly Increase (%)	6.0	2.5	2.5	4.0	0.5	2.0

Table 5.3: Values used to define the max hourly increase for each vital sign considered.

5.3.3 Results

For our test, we set TEPminer parameters as shown in Table 5.2 and Table 5.3. The first table contains values used independently from the vital sign considered, while the second one focus on the maximum hourly increase, in percentage, that we used to label a trend as {increasing,steady,decreasing}. This parameter is specific to every vital sign we used, that are: heart rate [HR], diastolic blood pressure [DBP], systolic blood pressure [SBP], respiratory rate [RR], body temperature [Temp], and oxygen saturation [SpO$_2$].

Overall, we analyze 144 different combination of parameters (the combination of 6 vital signs **x** 2 different max Δ_{start} **x** 2 max $\Delta_{duration}$**x** 6 Δ_y) and we obtained 10 822 440 *TE*-Ps in 906 seconds. Without multithread, the same mining took 1567 seconds and this shows the importance of parallelization in TEPminer.

Vital Sign	**Trend pre**	**Trend post**	**# Trends**
Diastolic Blood Pressure	STEADY	STEADY	121 190
	DECREASING	STEADY	14 336
	STEADY	INCREASING	11 910
Systolic Blood Pressure	STEADY	STEADY	123 014
	DECREASING	STEADY	13 171
	STEADY	INCREASING	12 531
Heart Rate	STEADY	STEADY	840 064
	DECREASING	STEADY	11 417
	STEADY	INCREASING	5923
Resp. Rate	STEADY	STEADY	71 490
	STEADY	INCREASING	3320
	INCREASING	STEADY	2896
SpO2	STEADY	STEADY	584 120
	STEADY	DECREASING	4740
	INCREASING	STEADY	3329
Temperature	STEADY	STEADY	38 864
	INCREASING	STEADY	14 018
	STEADY	INCREASING	2078

Table 5.4: Three most frequent types of *TE*-Ps for each vital sign with all the events considered together.

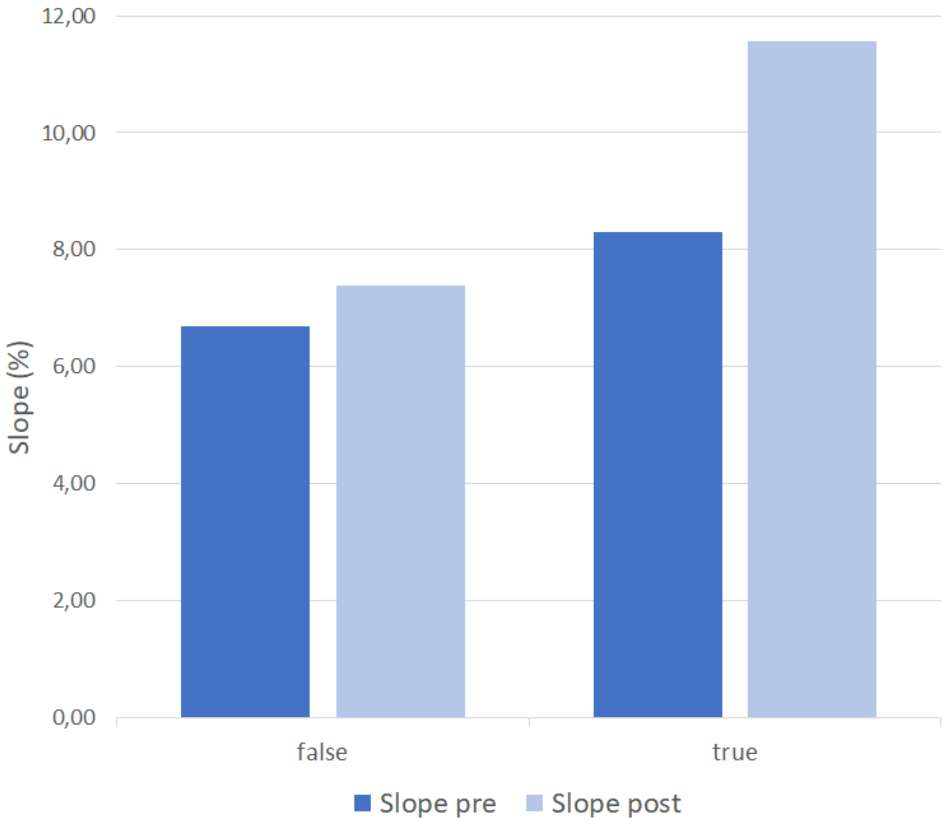


Figure 5-5: Given the same trend, there are slope differences when the event is actively influencing the trend. This figure shows the slope difference for an INCREASING trend related to heart rate, where the *true* flag indicates the events that are influencing the trends.

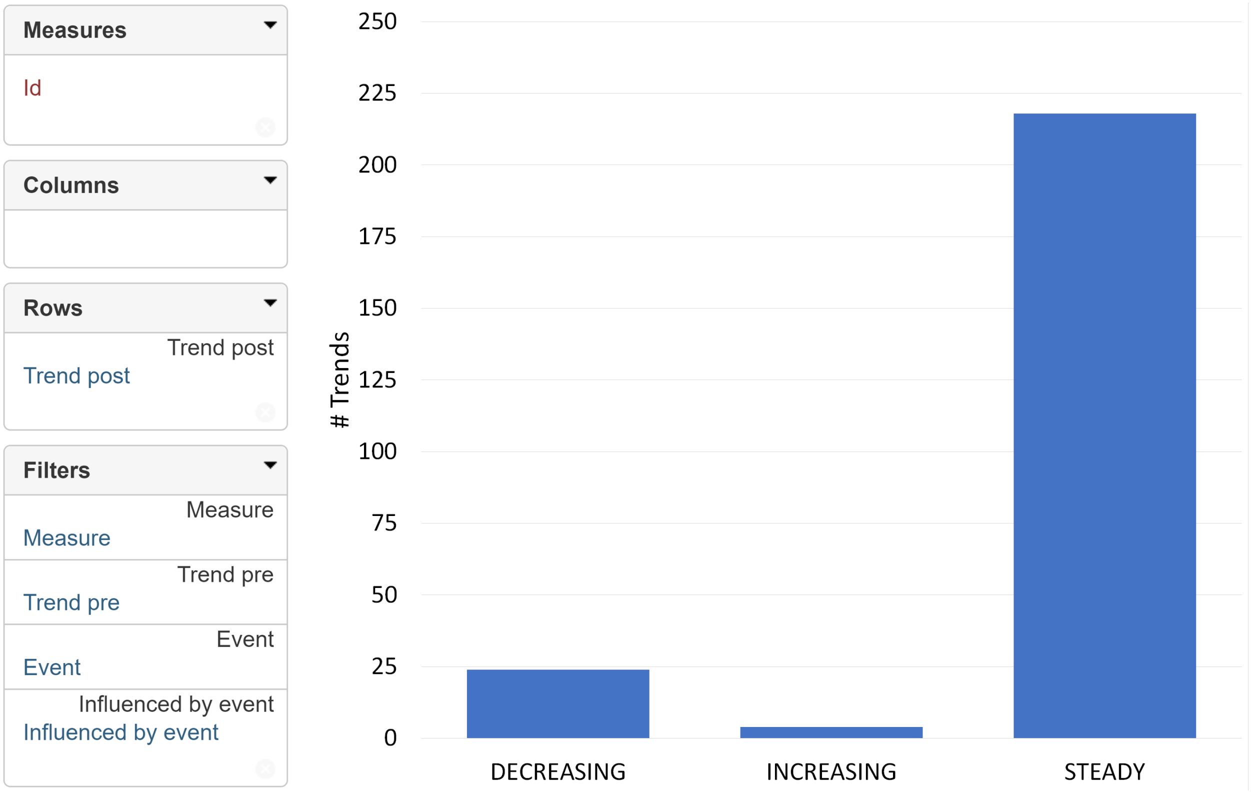
Table 5.4 shows the 3 most frequent kind of *TE*-Ps, where STEADY-STEADY represents the most frequent for any vital sign considered. An explanation could be that most of the events are not strongly influencing the considered vital sign, even if they change the slope of the trend. Apart from the STEADY-STEADY occurrence, all the other kind of *TE*-Ps could be intended as a process of stabilization of the vital signs, because they move from or to a STEADY position.

Moreover, TEPminer integrates a flag for every event, marking them only if they are actively influencing the trends or not (as explained in 5.1). The OLAP analysis allows us to analyze the average of slopes of a certain type of trend, and we expect a significantly difference between the slope that precedes an event and the one that succeeds it, when the event has an influence on the considered vital sign. Figure 5-5, that focus on INCREASING trends related to heart rate, shows exactly this difference in slopes.

Finally, Figure 5-6 shows an interesting example of multidimensional analysis. We focused on a specific drug, *propofol*, that is a strong sedative and hypnotic mainly used for the starting and maintenance of general anesthesia, sedation for mechanically ventilated adults, and procedural sedation; and its impact on an INCREASING respiratory rate.

In fact, we fixed *trend_pre* to be INCREASING in order to analyze the outcoming trends and it is clearly visible the effect of *propofol*. Overall, there are 248 *trend_post* and 218 of them (88%) indicate a stabilization in the respiratory rate, and in only 4 trends (1.6%) it is still increasing.

This example shows why multidimensional analysis are another method that helps us to confirm the effectiveness of a certain drug. Moreover, Figure 5-6 contains a column taken from the Saiku interface that shows the flexibility and ease of use to analyze our data cube. In fact, an user just need to select a subset of measures and dimensions, and arrange them in this column to perform the typical OLAP operations (e.g., *roll-up* or *drill-down*) and analysis.



Measures
Id

Columns

Rows
Trend post
Trend post

Filters
Measure
Measure
Trend pre
Trend pre
Event
Event
Influenced by event
Influenced by event

Figure 5-6: In this figure we show the effect of *propofol* on respiratory rate. We fixed the *trend_pre* as INCREASING and it's clearly visible the effect of *propofol* in stabilizing the respiratory rate. On the left, there is an excerpt of Saiku interface that underlines the flexibility of our OLAP analysis.

5.4 Conclusions

In this chapter we introduced a new kind of *Temporal Patterns*, called *Trend-Event Patterns* (*TE*-Ps), that are useful to infer correlations between trends and events in a completely automated way. We have evaluated and discussed their expressiveness by means of some examples taken from the clinical domain. More precisely, we proposed an algorithm to mine *TE*-Ps over a given dataset and we implemented it into a working prototype called *TEPminer*. We used *TEPminer* to mine data belonging to a healthcare domain of intensive care and in particular we applied our solution to the MIMIC-III EHR database. Our results prove *TEPminer* to be an interesting tool for mining clinical data. Moreover, *TE*-Ps were analyzed in a multidimensional aggregate way using an OLAP analysis.

In the future, we aim to deepen our understanding of *TE*-Ps through 4 different approaches. The first one focus on an extensive multidimensional analysis, where we will mix trends retrieved from different vital signs to observe the behavior of a certain drug on multiple parameters. An example could be to analyze the effect of some sedative, given after an INCREASING respiratory rate, on any other vital sign. Successively, it is interesting to extract association rules and functional dependencies, also in their approximate form, starting from *TE*-Ps. This will allow us to analyze *trend-event* rules from many different perspectives, any one of them with a different level of detail: we could analyze a single *TE*-P, perform a multidimensional OLAP analysis on an aggregation of *TE*-Ps, extract only *some* trend-event rules based on a certain support/confidence/etc, or even analyze trend-event rules from an attributes perspective. Finally, it is always fundamental to receive a validation on all these rules from clinicians.

Chapter 6

Mining Compact Predictive Pattern Sets Using Classification Model

In the previous chapter, we showed interesting temporal patterns extracted from ICU data contained in MIMIC III dataset. However, all the methodologies proposed in the previous chapters could extract rules that hold only for the given data. That is, if more data are added the mining has to be done again. The fact that these rules are not valid with additional data in the same dataset implies that the current rules are not capable of prediction.

In this chapter, we propose a new methodology to extract *predictive patterns*. This is done by studying new ways of improving pattern rule mining that can lead to a smaller set of rules that are sufficient to capture the essential underlying patterns in the data. This requires analyzing relations among the mined rules and defining criteria for assessing the importance of individual rules w.r.t. other rules. The key principle studied and applied in this work for filtering the rules is rule redundancy (i.e., eliminate spurious patterns). Briefly, a pattern is called spurious when it is predictive when evaluated alone, but is redundant given one of its subpatterns. Spurious patterns may be formed by adding irrelevant items to other simpler predictive patterns. Approach in [11] eliminates spurious patterns using statistical test based on binomial distribution. Later the same authors proposed a more robust Bayesian criterion to perform the spurious pattern elimination [9]. Our approach builds upon the minimum predictive pattern mining idea proposed above, in order to eliminate spurious and highly redundant rules, and attempts to improve it by reducing the set of mined minimum predictive rules using an auxiliary classification model that combines the rules into one model. Since, in general, the search for the optimal set of rules is equivalent to the optimal subset selection problem [63], we propose and experiment with a more efficient greedy rule selection algorithm that avoids the need to explore and evaluate all possible rules subsets.

Again, we have tested our method on data from MIMIC-III EHR dataset (that is illustrated in Section 2.3.1). More specifically, our goal is to discover patterns that are associated with sepsis and its treatments. We compare our method to the original one [11] and show that the number of rules found by our method is significantly smaller than the original set. Moreover, we show that the performance of the classification model that is based upon our rule set is close or better than classification models built by Batal's rule sets.

6.1 Method

6.1.1 Definitions

In this chapter we use some of the concepts that are already been described and defined in Section 2.1.5. In particular, the concept of Item (Definition 10) , Itemset pattern (Definition 12), Subpattern (Definition 13), Support(Definition 14), and Frequent Pattern(Definition 15).

Here we are interested in mining patterns that are predictive of class c. So for pattern P, we can define a predictive pattern (or a rule) R: $P \Rightarrow c$ with respect to class label c. The confidence of R is the precision (or posterior probability of c in group D_P). Note that confidence of $\Phi \Rightarrow c$ is the prior probability of c. We say that rule R': $P' \Rightarrow c'$ is a subrule of rule R: $P \Rightarrow c$ if $c' = c$ and $P' \subset P$.

Let $\Omega = \{P_1, ..., P_m\}$ be a set of patterns predictive of c. Given a dataset $D = \{\boldsymbol{x_i}, y_i\}_{i=1}^{n}$ defined in d-dimensional feature space and a set of patterns Ω the instances in D can be mapped into a new m-dimensional binary array D_Ω as follows:
$\boldsymbol{x_i} \rightarrow \{b_{i,1}, ..., b_{i,m}\}$ where $b_{i,j} = 1$ if $P_j \in \boldsymbol{x_i}$ and $b_{i,j} = 0$ if $P_j \notin \boldsymbol{x_i}$.
We refer to new $D_\Omega = \{\boldsymbol{x'_i}, y_i\}_{i=1}^{n}$ as to the pattern induced projection of the dataset D based on patterns in Ω. The pattern induced dataset D_Ω and its instances can be used to define and also learn a binary classification model $f : \boldsymbol{x'_i} \rightarrow y_i = c$ that distinguishes instances with the target class c from other classes. Effectively, this classification model combines a set of patterns predictive of c into a unified model for predicting the same class.

6.1.2 Problem

Our objective is to identify a small set of predictive patterns (rules) for the target class c from the data. To achieve this we propose a new two-step pattern mining process.

First, the number of predictive rules one can define by considering just the rule support and its precision can be enormous and may include a large number of spurious patterns. Hence we restrict our attention only to non-spurious rules. We mine these rules using Apriori algorithm proposed by [11] that includes binomial test when selecting more specific rules.

Second, to further limit the number of predictive rules we combine the minimal predictive patterns into a unified classification model to search for the optimal minimal pattern set Ω^* predictive of the target class c. We define the optimal pattern set to be the minimal pattern set that leads to the best combined generalization performance discriminating class c from the rest of the classes.

In the following we first describe the idea behind the minimum predictive patterns, and the unified classification models. After that we propose a greedy search algorithm that combines the two ideas into one search mechanism for identifying small sets of predictive patterns.

6.1.3 Minimum Predictive Patterns

Our solution builds upon the concept of minimum predictive patterns (MPPs) proposed by Batal and Hauskrecht [11].

Definition 22 (*Minimal Predictive Pattern*)**.** A predictive pattern $R : A \rightarrow c$ is called minimal, if and only if, R predicts class c significantly better than all its subpatterns.

The gist of this definition is that every item in the condition of the predictive pattern R is an important contributor to its prediction, that is, removal of any of the items in the condition would cause a significant drop in its predictive performance. The significance of the pattern R is determined using a statistical test derived from the binomial distribution. Let us assume we are interested in testing the significance of rule $R : A \rightarrow c$. Assume that pattern A consists of N instances, out of which N_c instances belong to class c. Let P_c represents the highest probability achieved by any subpattern of R, that is, $P_c = \max_{(} A' \subset A)Pr(c|A')$.

To test, if the pattern R is significantly different, we hypothesize (null hypothesis) that N_c is generated from N according to the binomial distribution with probability P_c. If we cannot reject the hypothesis at some significance level, then, R is not significantly different from the subpattern with P_c. However, we say that pattern R is significantly different when we can reject the above hypothesis and show that the probability that generated N_c class x instances out N is significantly higher than P_c. We can perform this test using a one sided significance test and calculate its p-value. If this p-value is significant (smaller than a significance level α), we conclude that R significantly improves the predictability of c over all its simplifications, and hence R is a MPP.

The mining algorithm to mine minimal predictive patterns relies on the Apriori algorithm that uses a minimum support parameter. The algorithm generates all patterns starting from more general patterns to more specific that satisfy the minimum support, but only the patterns that satisfy the binomial test (the minimality condition) are retained. As shown by studies in [11] such an algorithm retains significantly smaller subset of predictive patterns.

6.1.4 Combining Predictive Patterns via Classification Model

Our second solution attempts to reduce the number of minimum patters mined by considering their combinations. Briefly, we are interested in retaining only a subset of minimum predictive patterns that are critical for predictive performance of the classification model defined on the pattern induced dataset.

There are many classification models one can define on the binary dataset induced by the predictive patterns. In this work, instead of considering all possible classification models, we restrict our attention to linear support vector machines (SVM) models with shared discriminant functions (discriminating class c from the rest of the classes) that are defined by a linear combination of predictive patterns. To judge and compare the quality of such models across many features we use the area under the ROC curve (AUROC) statistic.

In general the problem of finding the optimal subset of minimum predictive patterns that leads to the best performing classification model is intractable. In order to make the search more efficient we resort to greedy pattern search approach. To make the choices of patterns we rely on the wrapper approach that tests, and selects patterns by considering the internal validation approach. That is, in order to compare two distinct sets of patterns Ω and Ω', we use the internal train and test splits of the data to evaluate the AUROC performance of the two sets in combination with the SVM model. The model and its patterns set with better AUROC performance is preferred. In the following we describe the specific algorithm we use to search a subset of minimum predictive patters to identify the best set.

6.1.5 Greedy Pattern Subset Selection Algorithm

Our approach starts by splitting dataset D into the training and test sets. All pattern selection and learning is always done on the training set. We use the test set only for the final evaluation.

Since our algorithm searches and compares many different subsets of predictive patterns, we use internal validation process to measure their quality and choose better subsets. Briefly, in order to evaluate and compare the goodness of a specific set of patterns Ω to other candidate sets, we use a classification model based on the linear SVM that is run on the data induced by Ω. We use multiple internal validation splits of the training data to make the comparison. The training dataset is divided as follows: first we randomly pick 30% of the data rows and use them as the test set, the remaining rows are reshuffled 10 times and for every reshuffle 80% of the data are used as the internal training set and the remaining 20% as the internal validation set. The goodness of Ω is then estimates by averaging the AUROC score for all internal splits obtained through reshuffling.

While our ultimate goal would be to find a set of predictive patterns that are optimal in terms of the quality of the predictive performance of a classifier that combines them, the full search is infeasible. To avoid the full pattern subset search, we adapt a greedy approach that generates, examines and selects the patterns level-wise, where a level k covers all k-patterns. More specifically, our method uses a two-stage procedure.

First, using an Apriori algorithm with the minimum support threshold and the binomial test proposed by Batal et al, we generate a set of minimum predictive patterns for each level k. Second, we use these minimum level-wise patterns to construct greedily the final set of patterns. We implemented two procedures to conduct the greedy search. One that searches and constructs the subset of patterns starting from the most general (level 1) patterns and gradually adds new more specific (higher level) patterns. We refer to this procedure as the top-bottom greedy procedure. The other procedure starts from the most specific patterns (the highest level minimum predictive patterns) and greedily adds to the set more general patterns of lower complexity. We refer to this method as to the bottom-up greedy procedure.

Let us assume that Ω' is our current set of patterns (selected in the previous steps). Our greedy search algorithm on level k works by first trying each minimum pattern on level k in combination with Ω'. Each of these combinations are ranked in terms of the AUROC score based on the internal validation. This order defines a greedy order in which all k-level minimum patterns are sequentially tried and if successful (in terms of AUROC improvement) they are added (one-by-one) to the resulting set of patterns. The same procedure for greedily adding the patterns on level k is applied whether we build the patterns in the top-down fashion (from level 1 patterns) or from the bottom-up (from highest level patterns). The reason for using the bottom-up greedy search process is that it tends to retain a greater number of the more specific patterns.

6.2 Experiments

6.2.1 Data

To test and validate our method, we analyze clinical data derived from MIMIC III dataset [58] with the goal of identifying patterns predictive of sepsis diagnosis. Further details on MIMIC are available in subsection 2.3.1. Before analysis, it is necessary to transform the MIMIC-III raw data in a form that we could mine. This was accomplished through an E.T.L. (Extract, Transform, Load) process. One source of our data was CHARTEVENTS, that is the vital signs table. We used it to extract specific measurements of heart rate, diastolic and systolic blood pressure, white blood cells, and body temperature across the admission. For each of these variables we created two attributes, one containing its maximum value during the hospitalization of a patient, and the other one containing its minimum value. Instead of numerical values, all these measurements were discretized to low, normal and medium ranges, using the thresholds shown in Table 6.1.

Other information we selected from the records came from PROCEDURES_ICD table which we used to determine whether a patient had a procedure or not (true\false attribute is created) during the hospitalization. We applied the same transformation to table DIAGNOSES_ICD to identify all the patients diagnosed with sepsis. INPUTEVENTS_MV table consists of medication administration records. We used it to extract some medications administered to the patient, such as vancomycin, piperacillin/tazobactam, ciprofloxacin, epinephrine, norepinephrine, vasopressin, dopamine, metoprolol, potassium chloride, phenylephrine, omeprazole (prilosec), and pantoprazole (protonix).

Let us note that while some of these medications are commonly used for treating patients with sepsis, whereas other medications such as metoprolol, potassium chloride, phenylephrine, omeprazole (prilosec), and pantoprazole (protonix) are more general. These were included to test the effectiveness of our method when mining patterns related to sepsis. At the end of the E.T.L. process we obtain data for 21 880 patients, 2806 of them with sepsis.

	Heart Rate	**Diastolic BP**	**Systolic BP**	**White Blood Cells**	**Body Temperature**
Low	< 60	< 60	< 90	< 4.0	< 36.0
High	> 90	> 90	> 140	> 12.0	> 38.0

Table 6.1: Thresholds used to discretize the considered vital signs in low, medium, high.

6.2.2 Results

Table 6.2 shows the results we obtained on MIMIC-III data for the minimum predictive rule mining approach by Batal and Hauskrecht [11], and two versions of our greedy classification model driven subset selection approach. The main statistics we use to evaluate the quality of the predictive rule set is the area under the ROC (Receiver Operating Characteristics) curve (AUROC) [42]. All AUROC statistics listed in the table are obtained on the test data. In addition to AUROC performances, we list the number of patterns found by the different methods. For example, the minimum predictive pattern (MPP) baseline used 62 patterns and reached AUROC performance of 0.8602. As we can see, both greedy methods outperformed (in terms of the AUROC classification performance) the baseline. Moreover this improvement is accompanied by a significant reduction in the total number of patterns used in the set compared to the baseline.

We note that while there is nearly no difference in the AUROC performance among the two versions of our greedy method, the number of patterns found and used by the two is significantly different. In particular, we observe that the majority of the patterns in the bottom-up approach are more complex patterns while the majority of patterns in the top-down approach are 1-patterns. This shows that bottom-up approach tends to keep more detailed patterns compared to the top-down approach.

Finally, we investigate for possible differences between the AUROC curves shown in Figure 6-1 (bottom-up approach) and in Figure 6-2. We used the method proposed by Delong et al. [38] to calculate the standard error of the AUROC curves and also to find differences between the two AUROC. Let us recall that an area of 0.8643 means that a randomly selected individual from the positive group has a test value larger than that for a randomly chosen individual from the negative group in 86.43% of the time, whereas the 95% confidence interval is the interval in which the true (population) Area under the ROC curve lies with 95% confidence.

The significance level P shows the probability of the hypothesis that the difference between the two AUROC curves is 0, while z statistic measures standard deviation and helps to decide whether or not to reject the null hypothesis. For example, the z score values when using a 95% confidence interval are ± 1.96 standard deviations and P is 0.05. If z score is between ± 1.96, P will be > 0.05, and the null hypothesis cannot be reject. Otherwise, if the Z score falls outside that range, P will be < 0.05 and it is possible to reject the null hypothesis, considering a significant difference between the two distributions.

The summarization of the differences between the two curves is shown in Figure 6-3. In all these Figures (6-1, 6-2, 6-3) the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.

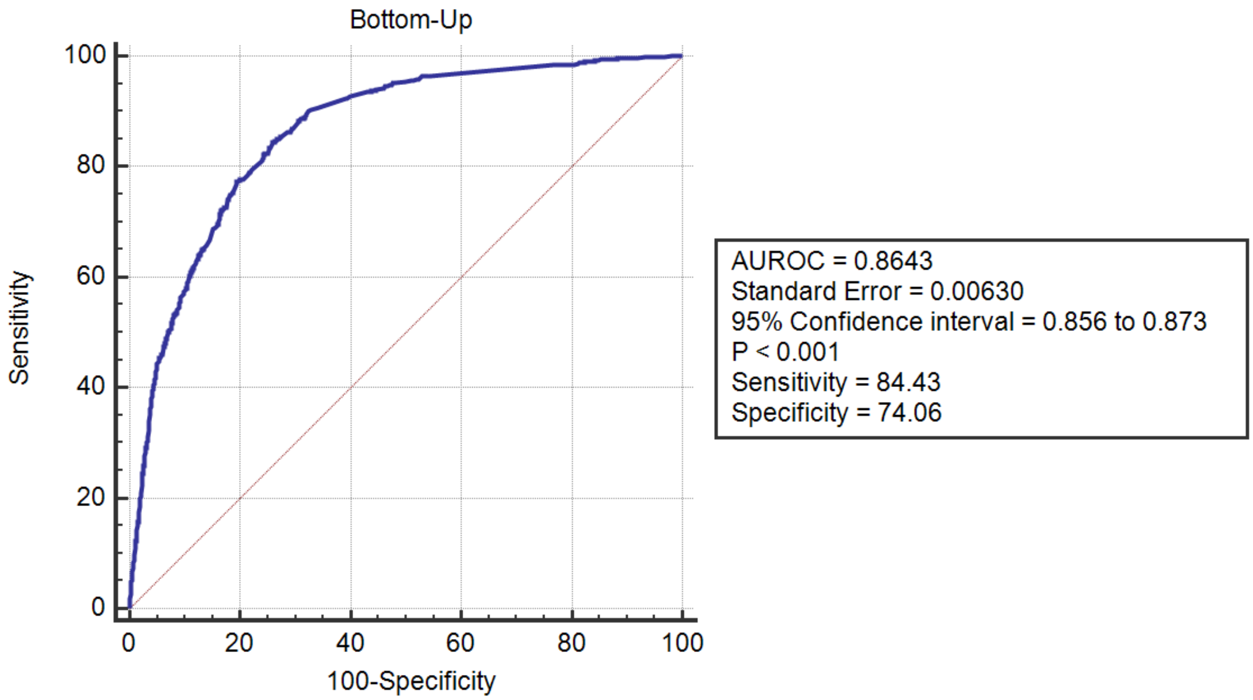


Figure 6-1: ROC curve for our bottom-up approach.

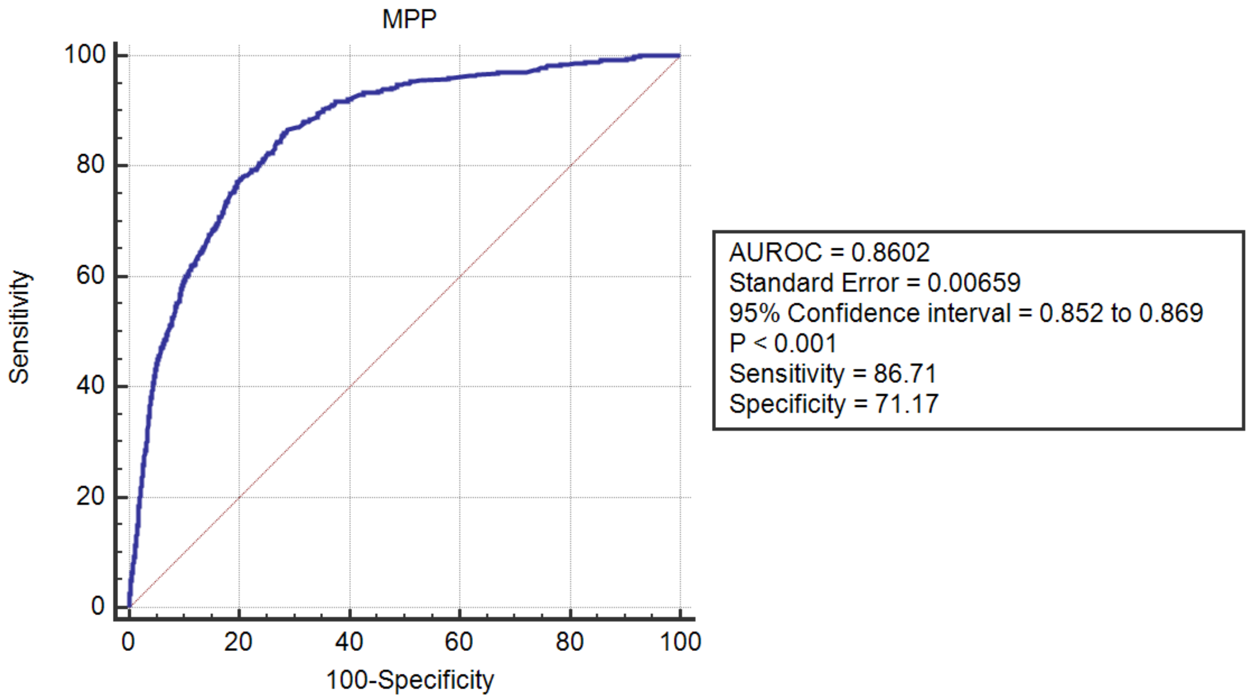


Figure 6-2: ROC curve for Batal's MPP approach.

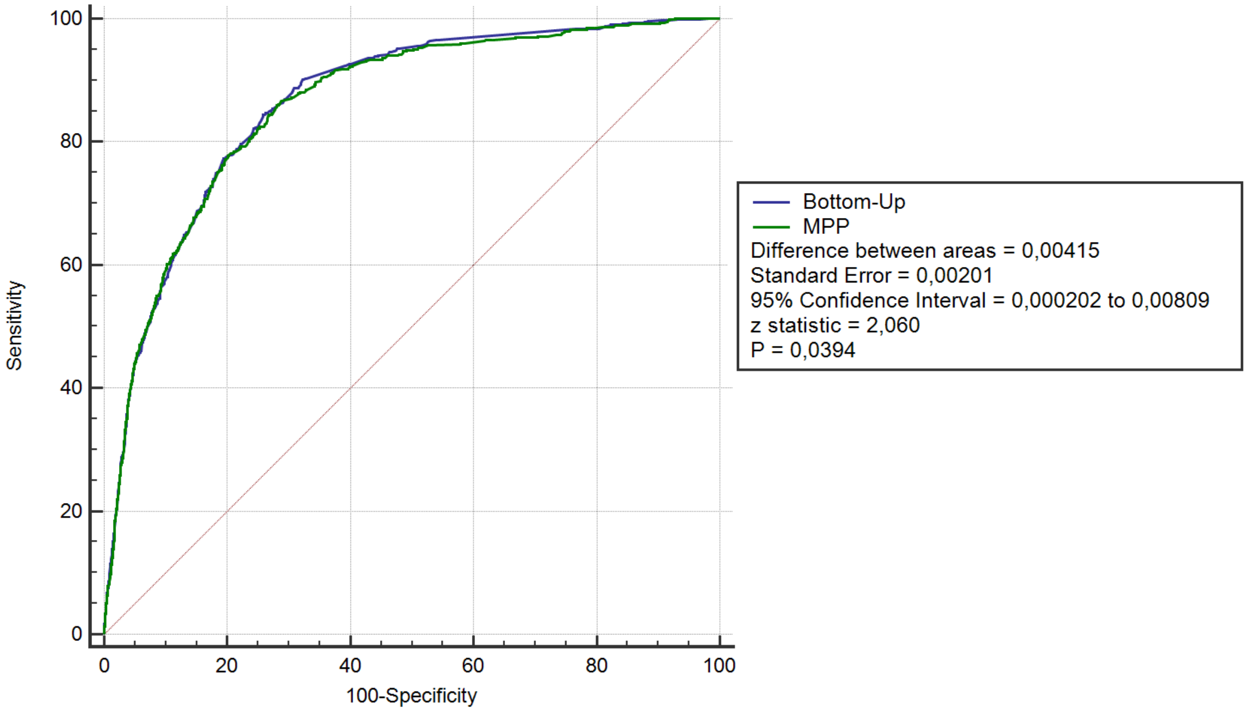


Figure 6-3: Comparison between MPP and our bottom-up ROC curves to show the differences.

Method	AUROC	Number of patterns
MPP (Batal et al)	0.8602	62
Our method (bottom-up)	0.8643	33
Our method (top-down)	0.8635	19

Table 6.2: Comparison between the results for our method and Batal et al's predictive pattern mining method.

6.3 Discussion

Sepsis is the systemic response to infection, and there are many conditions that would indicate its occurrence during the admission or hospital stay, such as: temperature $> 38\,°C$ or $< 36\,°C$; heart rate > 90 beats per minute; systolic blood pressure < 90 mm Hg, and white blood cell count $>$ 12,000/cu mm or $<$ 4,000/cu mm [18]. Moreover, patients with sepsis are usually treated with antibiotics such as vancomycin, piperacillin / tazobactam, ciprofloxacin, and drugs treating episodes of hypotension such as epinephrine, norepinephrine, phenylephrine, vasopressin, and dopamine [37].

Table 6.3 lists all minimal predictive patterns that we mined using the bottom-up greedy procedure. The table entries include the absolute weight the rule was assigned by the final classification model, the rule support and the rule precision. By analyzing the results with

Pattern	**Rule Weight**	**Support**	**Precision**
Norepinephrine = true	0.4667	0.1453	0.4933
Norepinephrine = true & Vancomycin = true	0.2628	0.1114	0.5615
Piperacillin/Tazobactam = true	0.2476	0.143	0.3991
MaxSystolicBloodPressure = low	0.1981	0.4311	0.1571
Ciprofloxacin = true	0.1750	0.1157	0.2970
Vancomycin = true	0.1619	0.3890	0.2612
Pantoprazole(Protonix) = true & MaxSystolicBloodPressure = low	0.1418	0.1437	0.2801
Norepinephrine = true & Piperacillin/Tazobactam = true	0.1159	0.0566	0.6482
MaxWhiteBloodCells = high	0.1017	0.5669	0.1699
MinWhiteBloodCells = low & MaxHeartRate = high	0.0870	0.0653	0.2780
Vancomycin = true & MinWhiteBloodCells = high	0.0856	0.0940	0.2773
PotassiumChloride = true & MaxWhiteBloodCells = high	0.0738	0.3275	0.1793
Vancomycin = true & MaxHeartRate = high	0.0628	0.3015	0.2871
MinWhiteBloodCells = low	0.0601	0.0913	0.2312
MinDiastolicBloodPressure = low & MaxWhiteBloodCells = high	0.0533	0.3102	0.1847
Vancomycin = true & MaxWhiteBloodCells = high	0.0527	0.2821	0.2797
Pantoprazole(Protonix) = true & Piperacillin/Tazobactam = true	0.0512	0.0662	0.4438
MaxWhiteBloodCells = high & MaxSystolicBloodPressure = low	0.0500	0.3255	0.1792
Piperacillin/Tazobactam = true & MaxWhiteBloodCells = high	0.0471	0.1106	0.4238
Pantoprazole(Protonix) = true & Ciprofloxacin = true	0.0380	0.0574	0.3436
MinTemp = low	0.0369	0.0618	0.1706
Vancomycin = true & *MaxDiastolicBloodPressure = high*	0.0324	0.1089	0.3023
Ciprofloxacin = true & MaxHeartRate = high	0.0241	0.0930	0.3251
Ciprofloxacin = true & MaxWhiteBloodCells = high	0.0117	0.0873	0.3263
Pantoprazole(Protonix) = true & Norepinephrine = true	0.0094	0.0679	0.5496
MaxWhiteBloodCells = high & MaxHeartRate = high	0.0075	0.4129	0.1974
Pantoprazole(Protonix) = true & Vancomycin = true	0.0058	0.1433	0.3423
Pantoprazole(Protonix) = true & *PotassiumChloride = true*	0.0053	0.1616	0.2442
Pantoprazole(Protonix) = true & MinDiastolicBloodPressure = low	0.0047	0.1369	0.2882
PotassiumChloride = true & MaxDiastolicBloodPressure = high	0.0047	0.1320	0.2184
MaxDiastolicBloodPressure = high & MaxHeartRate = high	0.0001	0.1451	0.2278
MaxSystolicBloodPressure = low & MaxHeartRate = high	0.0015	0.3109	0.1942
Pantoprazole(Protonix) = true & MaxWhiteBloodCells = high	0.0001	0.1851	0.2534

Table 6.3: The mined set of minimal predictive patterns with their absolute weight, support and precision.

respect to sepsis symptoms and treatments we see 21 patterns (out of 33) that match exactly sepsis related symptoms and/or treatments, and 9 more with the sepsis related patterns but in conjunction with Pantoprazole (Protonix). Pantoprazole is a proton pump inhibitor (PPI) and, even though it is not used to treat sepsis, PPIs are used for stress-related mucosal damage (SRMD). SRMD is an erosive gastritis of unclear pathophysiology, which can occur rapidly after a severe insult such as trauma, surgery, *sepsis* or burns [20]. In other words, it is still reasonable to mine patterns with Pantoprazole, because it is weakly related to sepsis. Finally, we have only 3 patterns, indicated in Table 6.3 in italic, that include items we would consider to be weakly related to sepsis: 2 patterns have $MaxDiastolicBloodPressure = high$ and one that includes $PotassiumChloride = true$. This demonstrates our algorithm is able to select a much smaller subset of patterns compared to MPP method and that the majority of the patterns predictive of sepsis are reasonable.

6.4 Conclusion

In this work we have developed and tested a new framework for mining predictive patterns that compactly describe a class of interest. It uses a greedy algorithm to mine the most predictive patterns level-wise and including only those that improve the overall class prediction performance. We tested our approach on intensive care data from MIMIC-III EHR database, focusing on patterns predictive of sepsis. The results preserve the overall classification quality of state-of-the-art methods based on minimum predictive pattern mining approach, but with a significant reduction in the number of extracted patterns.

Chapter 7

Conclusions

In this thesis, we proposed a framework (graphically depicted in Figure 1-1) that integrates different approximate data mining techniques applied to clinical data. The framework includes and extends different mining techniques and, in particular, it focuses on *Approximate Temporal Functional Dependencies* and *Patterns*. For the first one, two different proposals have been done: *Pure Temporally Evolving Functional Dependencies* (Chapter 3) and *Multi Approximate Temporal Functional Dependencies* (Chapter 4). Even for patterns, we proposed two new different mining techniques: *Trend-Event Patterns* (Chapter 5) and *Predictive Patterns* (Chapter 6).

The flexibility and interestingness of this framework ares given by different kinds of information it can extract even from the same clinical dataset. Chapters 4, 5, and 6 provide some concrete examples of these features. Using ICU data contained in the publicly-available MIMIC III dataset, we were able to extract a specific kind of *Approximate Temporal Functional Dependency*, *Temporal Pattern*, or *Predictive Pattern*.

The flexibility of this framework is extended to the underlying tools used to mine clinical data. Indeed, it is not necessary to create a new tool for every new technique proposed, but it is quite straightforward to extend the current framework.

Some limitations of the current framework need to be discussed and faced. From the computational side, when we have to combine many different attributes in a powerset, we inherently have to deal with an exponential complexity ($\mathcal{O}(2^n)$ - where n is the number of attributes). Nevertheless, pruning strategies may help to reduce the real number of combinations to test. This, combined with parallelization techniques, may help to reduce the computational time. Often this is still not enough, and we have to limit our set of attributes to retrieve results in a reasonable time. Moreover, the framework needs further clinical validations. In our previous discussions with clinicians, we noted that it is challenging to have insightful feedback on the finer details, such as tuning the length of a sliding window for *MAT*-FDs or tuning the parameters to retrieve *TE*-Ps. This should encourage us to increase the effort in creating and refining techniques that are also easily adoptable by clinicians.

Overall, clinicians gave us positive feedback on the results obtained. In particular, both *AT*-FDs generated some interest because their rules are valid at the attribute-level, whereas usually they have to deal with rules that explicitly consider attribute values. Attribute-level rules may help them to validate or discover concepts at a higher level of abstraction or aggregation. On the other hand, pattern mining produces results clinicians are more familiar with. Even though *TE*-Ps were initially used only for validation, they result interesting when used in their aggregate form. *TE*-Psare not only useful to validate known behaviors, but they also help to analyze the amount of outliers and their behavior.

Predictive patterns share the same advantages of *TE*-Ps. Moreover, they also have the benefit of prediction, but, for now, they are not considering the temporal aspect. They have been successfully used to verify the clinical conditions that affect a patient with severe sepsis, which is a disease hard to diagnose because many of its symptoms are common to other diseases.

Right now, the framework includes techniques to mine *temporal patterns* and *predictive patterns*. In the future, it would be interesting to combine these two concepts and discover some new interesting *predictive temporal patterns*. This idea could possibly develop in two different directions.

The first one is to use the classification method seen in Chapter 6 and use it to predict a set of the most “meaningful” temporal patterns previously discovered. Let us recall that with the current Trend-Event Patterns we have hundreds of thousands of patterns, and right now, the best way to analyze them is in some aggregate form through OLAP analysis. However, the combination of a classifier and temporal patterns could lead to a small (but highly significant) set of patterns.

The second direction consists of the integration of the *temporal* aspect in the current *predictive patterns*. In Chapter 6, we extracted patterns that predict sepsis. However, it could be interesting to understand whether these patterns change during time (e.g., if they hold within a fixed-size sliding window).

Both directions could help us to obtain an even smaller and more descriptive set of patterns, and this is particularly helpful in the path of introducing these patterns in a daily clinical routine.

Bibliography

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 94–105. ACM Press, 1998. doi:10.1145/276304.276314.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993.*, pages 207–216. ACM Press, 1993. doi:10.1145/170035.170072.

[3] Rakesh Agrawal and John C. Shafer. Parallel mining of association rules. *IEEE Trans. Knowl. Data Eng.*, 8(6):962–969, 1996. doi:10.1109/69.553164.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of VLDB*, 1994. URL: https://dl.acm.org/doi/10.5555/645920.672836.

[5] F Amaddeo, G Bisoffi, R Micciolo, M Piccinelli, and M Tansella. Frequency of contact with community-based psychiatric services and the lunar cycle: a 10-year case-register study. *Social psychiatry and psychiatric epidemiology*, 32(6):323–326, 1997. doi:10.1007/BF00805436.

[6] Francesco Amaddeo, Jennifer Beecham, Paola Bonizzato, A Fenyo, M Knapp, and Michele Tansella. The use of a case register to evaluate the costs of psychiatric care. *Acta Psychiatrica Scandinavica*, 95(3):189–198, 1997. doi:10.1111/J.1600-0447.1997.TB09619.X.

[7] Francesco Amaddeo and Michele Tansella. Information systems for mental health. *Epidemiology and Psychiatric Sciences*, 18(1):1–4, 2009. doi:10.1017/S1121189X00001378.

[8] R. C. Barr, S. M. Blanchard, and D. A. Dipersio. Sapa-2 is the fan. *IEEE Transactions on Biomedical Engineering*, BME-32(5):337–337, May 1985. doi:10.1109/TBME.1985.325548.

[9] Iyad Batal, Gregory Cooper, and Milos Hauskrecht. A Bayesian Scoring Technique for Mining Predictive and Non-Spurious Rules. In *Proceedings of the European conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, 2012. doi:10.1007/978-3-642-33486-3_17.

[10] Iyad Batal and Milos Hauskrecht. A supervised time series feature extraction technique using DCT and DWT. In M. Arif Wani, Mehmed M. Kantardzic, Vasile Palade, Lukasz A. Kurgan, and Yuan (Alan) Qi, editors, *International Conference on Machine Learning and Applications, ICMLA 2009, Miami Beach, Florida, USA, December 13-15, 2009*, pages 735–739. IEEE Computer Society, 2009. URL: https://ieeexplore.ieee.org/xpl/conhome/5379696/proceeding, doi:10.1109/ICMLA.2009.13.

[11] Iyad Batal and Milos Hauskrecht. Constructing classification features using minimal predictive patterns. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 869–878. ACM, 2010. doi:10.1145/1871437.1871549.

[12] Iyad Batal, Hamed Valizadegan, Gregory F. Cooper, and Milos Hauskrecht. A pattern mining approach for classifying multivariate temporal data. In Fang-Xiang Wu, Mohammed Javeed Zaki, Shinichi Morishita, Yi Pan, Stephen Wong, Anastasia Christianson, and Xiaohua Hu, editors, *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2011, Atlanta, GA, USA, November 12-15, , 2011*, pages 358–365. IEEE Computer Society, 2011. URL: https://ieeexplore.ieee.org/xpl/conhome/6120121/proceeding, doi:10.1109/BIBM.2011.39.

[13] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 436–442. ACM, 2002. doi:10.1145/775047.775110.

[14] Riccardo Bellazzi, Cristiana Larizza, Paolo Magni, and Roberto Bellazzi. Temporal data mining for the quality assessment of hemodialysis services. *Artificial Intelligence in Medicine*, 34(1):25 – 39, 2005. Artificial Intelligence in Medicine in Europe {AIME} '03. URL: http://www.sciencedirect.com/science/article/pii/S093336570400123X, doi:10.1016/j.artmed.2004.07.010.

[15] Riccardo Bellazzi, Cristiana Larizza, and Alberto Riva. Temporal abstractions for interpreting diabetic patients monitoring data. *Intelligent Data Analysis*, 2(1-4):97–122, 1998. doi:10.1016/S1088-467X(98)00020-1.

[16] Riccardo Bellazzi and Blaz Zupan. Predictive data mining in clinical medicine: current issues and guidelines. *International journal of medical informatics*, 77(2):81–97, 2008. doi:10.1016/J.IJMEDINF.2006.11.006.

[17] M. Bertinelli, A. Castelli, C. Combi, and F. Pinciroli. Data compression applied to dynamic electrocardiography. *Medical and Biological Engineering and Computing*, 27(1):33–40, Jan 1989. `doi:10.1007/BF02442167`.

[18] Roger C Bone, Robert A Balk, Frank B Cerra, R Phillip Dellinger, Alan M Fein, William A Knaus, Roland MH Schein, and William J Sibbald. Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis. *Chest*, 101(6):1644–1655, 1992. `doi:10.1378/CHEST.101.6.1644`.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series, 1984. ISBN-10: 0534980538.

[20] Stephen Brett. Science review: the use of proton pump inhibitors for gastric acid suppression in critical illness. *Critical care*, 9(1):45, 2004. `doi:10.1186/CC2980`.

[21] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986. `doi:10.1109/TC.1986.1676819`.

[22] S. Chakravarty and Y. Shahar. Acquisition and analysis of repeating patterns in time-oriented clinical data. *Methods of information in medicine*, 40(5):410–420, 2001. `doi:10.1055/S-0038-1634201`.

[23] Shubha Chakravarty and Yuval Shahar. CAPSUL: A constraint-based specification of repeating patterns in time-oriented data. *Ann. Math. Artif. Intell.*, 30(1-4):3–22, 2000. `doi:10.1023/A:1016661915959`.

[24] Dustin Charles, Meghan Gabriel, Talisha Searcy, et al. Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2013. *ONC data brief*, 9:1–9, 2013. URL: `https://www.healthit.gov/sites/default/files/oncdatabrief16.pdf`.

[25] Hong Cheng, Xifeng Yan, Jiawei Han, and Chih-Wei Hsu. Discriminative frequent pattern analysis for effective classification. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 716–725. IEEE Computer Society, 2007. URL: `https://ieeexplore.ieee.org/xpl/conhome/4221634/proceeding`, `doi:10.1109/ICDE.2007.367917`.

[26] Edgar Frank Codd. Normalized data structure: A brief tutorial. In Edgar Fred Codd and Albert L. Dean, editors, *SIGFIDET Workshop*, pages 1–17. ACM, 1971. `doi:10.1145/1734714.1734716`.

[27] Carlo Combi and Luca Chittaro. Abstraction on clinical data sequences: an object-oriented data model and a query language based on the event calculus. *Artificial Intelligence in Medicine*, 17(3):271–301, 1999. `doi:10.1016/S0933-3657(99)00022-6`.

[28] Carlo Combi, Massimo Franceschet, and Adriano Peron. Representing and reasoning about temporal granularities. *J. Log. Comput.*, 14(1):51–77, 2004. doi:10.1093/logcom/14.1.51.

[29] Carlo Combi, Matteo Mantovani, Alberto Sabaini, Pietro Sala, Francesco Amaddeo, Ugo Moretti, and Giuseppe Pozzi. Mining approximate temporal functional dependencies with pure temporal grouping in clinical databases. *Comp. in Bio. and Med.*, 62:306–324, 2015. doi:10.1016/j.compbiomed.2014.08.004.

[30] Carlo Combi, Matteo Mantovani, and Pietro Sala. Discovering quantitative temporal functional dependencies on clinical data. In *2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, August 23-26, 2017*, pages 248–257. IEEE Computer Society, 2017. URL: https://ieeexplore.ieee.org/xpl/conhome/8030514/proceeding, doi:10.1109/ICHI.2017.80.

[31] Carlo Combi, Angelo Montanari, and Giuseppe Pozzi. The T4SQL Temporal Query Language. In *CIKM*, pages 193–202. ACM, 2007. doi:10.1145/1321440.1321470.

[32] Carlo Combi, Angelo Montanari, and Pietro Sala. A uniform framework for temporal functional dependencies with multiple granularities. In *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Proceedings*, pages 404–421, 2011. doi:10.1007/978-3-642-22922-0_24.

[33] Carlo Combi, Romeo Rizzi, and Pietro Sala. The price of evolution in temporal databases. In *22nd International Symposium on Temporal Representation and Reasoning, TIME 2015*, pages 47–58. IEEE, 2015. doi:10.1109/TIME.2015.24.

[34] Carlo Combi and Alberto Sabaini. Extraction, analysis, and visualization of temporal association rules from interval-based clinical data. In *Artificial Intelligence in Medicine - 14th Conference on Artificial Intelligence in Medicine, AIME 2013. Proceedings*, pages 238–247, 2013. doi:10.1007/978-3-642-38326-7_35.

[35] Carlo Combi and Pietro Sala. Mining approximate interval-based temporal dependencies. *Acta Inf.*, 53(6-8):547–585, 2016. doi:10.1007/s00236-015-0246-x.

[36] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi:10.1007/BF00994018.

[37] R Phillip Dellinger, Mitchell M Levy, Andrew Rhodes, Djillali Annane, Herwig Gerlach, Steven M Opal, Jonathan E Sevransky, Charles L Sprung, Ivor S Douglas, Roman Jaeschke, et al. Surviving sepsis campaign: international guidelines for management of severe sepsis and septic shock, 2012. *Intensive care medicine*, 39(2):165–228, 2013. doi:10.1097/CCM.0B013E31827E83AF.

[38] ER DeLong, DM DeLong, and DL Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 44(3):837—845, September 1988. doi:10.2307/2531595.

[39] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17:1036–1050, 2005. doi:10.1109/TKDE.2005.127.

[40] Marián Dvorský. Common permutation problem. *CoRR*, abs/0803.4261, 2008. URL: http://arxiv.org/abs/0803.4261.

[41] I Ralph Edwards and Cecilia Biriell. Harmonisation in pharmacovigilance. *Drug safety*, 10(2):93–102, 1994. doi:10.2165/00002018-199410020-00001.

[42] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006. doi:10.1016/J.PATREC.2005.10.010.

[43] National Center for Health Statistics (US). *The International Classification of Diseases, 9th Revision, Clinical Modification: Procedures: tabular list and alphabetic index*, volume 3. US Department of Health and Human Services, Public Health Service, Health Care Financing Administration, 1980. URL: https://www.cdc.gov/nchs/icd/icd9cm.htm.

[44] Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006. doi:10.1145/1132960.1132963.

[45] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. doi:10.1161/01.cir.101.23.e215.

[46] L Grigoletti, G Perini, A Rossi, A Biggeri, C Barbui, M Tansella, and F Amaddeo. Mortality and cause of death among psychiatric patients: a 20-year case-register study in an area with a community-based system of care. *Psychological medicine*, 39(11):1875–1884, 2009. doi:10.1017/S0033291709005790.

[47] Birger Haarbrandt, Erik Tute, and Michael Marschollek. Automated population of an i2b2 clinical data warehouse from an openehr-based data repository. *Journal of Biomedical Informatics*, 63:277–294, 2016. doi:10.1016/j.jbi.2016.08.007.

[48] Ira J Haimowitz and Isaac S Kohane. Managing temporal worlds for medical trend diagnosis. *Artificial Intelligence in Medicine*, 8(3):299–321, 1996. doi:10.1016/0933-3657(95)00037-2.

[49] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011. eBook ISBN: 9780123814807.

[50] David T. Hau and Enrico W. Coiera. Learning qualitative models of dynamic systems. *Machine Learning*, 26(2):177–211, 1997. doi:10.1023/A:1007317323969.

[51] Manfred Hauben and Lester Reich. Communication of findings in pharmacovigilance: use of the term “signal” and the need for precision in its use. *European journal of clinical pharmacology*, 61(5-6):479–480, 2005. `doi:10.1007/S00228-005-0951-4`.

[52] John H. Holmes, Thomas E. Elliott, Jeffrey S. Brown, Marsha A. Raebel, Arthur J. Davidson, Andrew F. Nelson, Annie Chung, Pierre La Chance, and John F. Steiner. Clinical research data warehouse governance for distributed research networks in the USA: a systematic review of the literature. *JAMIA*, 21(4):730–736, 2014. `doi:10.1136/amiajnl-2013-002370`.

[53] Frank Höppner. *Knowledge discovery from sequential data*. PhD thesis, Braunschweig University of Technology, Germany, 2003. URL: `http://opus.tu-bs.de/opus/volltexte/2003/406/index.html`.

[54] Frank Höppner and Frank Klawonn. *Finding Informative Rules in Interval Sequences*, pages 125–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. `doi:10.1007/3-540-44816-0_13`.

[55] Ykä Huhtala, Juha Karkkainen, Pasi Porkka, and Hannu Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 392–401. IEEE, 1998. `doi:10.1109/ICDE.1998.655802`.

[56] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999. `doi:10.1093/COMJNL/42.2.100`.

[57] Christian S. Jensen, Richard T. Snodgrass, and Michael D. Soo. Extending existing dependency theory to temporal databases. *IEEE Trans. Knowl. Data Eng.*, 8(4):563–582, 1996. `doi:10.1109/69.536250`.

[58] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016. `doi:10.1038/sdata.2016.35`.

[59] Viktor Jovanoski and Nada Lavrac. Classification rule learning with apriori-c. In *EPIA*, 2001. `doi:10.1007/3-540-45329-6_8`.

[60] Po-shan Kam and Ada Wai-Chee Fu. Discovering temporal patterns for interval-based events. In Yahiko Kambayashi, Mukesh K. Mohania, and A Min Tjoa, editors, *Data Warehousing and Knowledge Discovery, Second International Conference, DaWaK 2000, London, UK, September 4-6, 2000, Proceedings*, volume 1874 of *Lecture Notes in Computer Science*, pages 317–326. Springer, 2000. `doi:10.1007/3-540-44466-1\_32`.

[61] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2002. URL: `https://dl.acm.org/doi/book/10.5555/560521`.

[62] Jyrki Kivinen and Heikki Mannila. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149, 1995. `doi:10.1016/0304-3975(95)00028-U`.

[63] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996. URL: `https://dl.acm.org/doi/10.5555/3091696.3091731`.

[64] Benjamin Kuipers. Qualitative simulation. *Artificial intelligence*, 29(3):289–338, 1986. `doi:10.1016/0004-3702(86)90073-1`.

[65] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings of ICDM*, 2001. `doi:10.1109/ICDM.2001.989534`.

[66] Nada Lavrač. Selected techniques for data mining in medicine. *Artificial intelligence in medicine*, 16(1):3–23, 1999. `doi:10.1016/S0933-3657(98)00062-1`.

[67] Jørn Lind-Nielsen. *BuDDy - A Binary Decision Diagram Package*. `http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.itu.dk/research/buddy/`.

[68] Chuanren Liu, Kai Zhang, Hui Xiong, Geoff Jiang, and Qiang Yang. Temporal skeletonization on sequential data: patterns, categorization, and visualization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 13361345. ACM, 2014. `doi:10.1109/TKDE.2015.2468715`.

[69] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Functional and approximate dependency mining: database and fca points of view. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):93–114, 2002. `doi:10.1080/09528130210164143`.

[70] Riccardo Lora, Alberto Sabaini, Carlo Combi, and Ugo Moretti. Designing the reconciled schema for a pharmacovigilance data warehouse through a temporally-enhanced er model. In *Proceedings of the 2012 international workshop on Smart health and wellbeing*, pages 17–24. ACM, 2012. `doi:10.1145/2389707.2389711`.

[71] Michael Mampaey, Nikolaj Tatti, and Jilles Vreeken. Tell me what i need to know: succinctly summarizing data with itemsets. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 573–581. ACM, 2011. `doi:10.1145/2020408.2020499`.

[72] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, 1997. `doi:10.1023/A:1009748302351`.

[73] Matteo Mantovani. Approximate temporal functional dependencies on clinical data. In *2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, August 23-26, 2017*, page 328. IEEE Computer Society, 2017. URL: https://ieeexplore.ieee.org/xpl/conhome/8030514/proceeding, doi:10.1109/ICHI.2017.30.

[74] Matteo Mantovani, Carlo Combi, and Milos Hauskrecht. Mining compact predictive pattern sets using classification model. In David Riaño, Szymon Wilk, and Annette ten Teije, editors, *Artificial Intelligence in Medicine - 17th Conference on Artificial Intelligence in Medicine, AIME 2019, Poznan, Poland, June 26-29, 2019, Proceedings*, volume 11526 of *Lecture Notes in Computer Science*, pages 386–396. Springer, 2019. doi:10.1007/978-3-030-21642-9_49.

[75] Matteo Mantovani, Carlo Combi, and Matteo Zeggiotti. Discovering and analyzing trend-event patterns on clinical data. In *2019 IEEE International Conference on Healthcare Informatics, ICHI 2019, Xi'an, China, June 10-13, 2019*, pages 1–10. IEEE, 2019. URL: https://ieeexplore.ieee.org/xpl/conhome/8895688/proceeding, doi:10.1109/ICHI.2019.8904774.

[76] MedDRA MSSO. About MedDRA. In *About MedDRA*, pages –, 2010. URL: http://www.meddramsso.com/public_about_meddra.asp.

[77] R.H.B. Meyboom, M. Lindquist, A.C.G. Egberts, and I.R. Edwards. Signal selection and follow-up in pharmacovigilance. *Drug Safety*, 25(6):459–465, 2002. doi:10.2165/00002018-200225060-00011.

[78] Fabian Mörchen. Algorithms for time series knowledge mining. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 668–673. ACM, 2006. doi:10.1145/1150402.1150485.

[79] Robert Moskovitch and Yuval Shahar. Medical temporal-knowledge discovery via temporal abstraction. In *AMIA annual symposium proceedings*, volume 2009, page 452. American Medical Informatics Association, 2009. PMID:20351898.

[80] Robert Moskovitch and Yuval Shahar. Classification of multivariate time series via temporal abstraction and time intervals mining. *Knowl. Inf. Syst.*, 45(1):35–74, 2015. doi:10.1007/s10115-014-0784-5.

[81] G Niklas Norén, Roland Orre, Andrew Bate, and I Ralph Edwards. Duplicate detection in adverse drug reaction surveillance. *Data Mining and Knowledge Discovery*, 14(3):305–328, 2007. doi:10.1007/s10618-006-0052-8.

[82] Object Management Group. Business Process Model and Notation (BPMN), v2.0.2. URL: http://www.omg.org/spec/BPMN/2.0.2/PDF/.

[83] World Health Organization. The importance of pharmacovigilance, 2002.

[84] World Health Organization et al. *The ICD-10 classification of mental and behavioural disorders: clinical descriptions and diagnostic guidelines.* Geneva: World Health Organization, 1992.

[85] Panagiotis Papapetrou, George Kollios, Stan Sclaroff, and Dimitrios Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 354–361. IEEE Computer Society, 2005. URL: `https://ieeexplore.ieee.org/xpl/conhome/10470/proceeding`, `doi:10.1109/ICDM.2005.50`.

[86] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan,: mining sequential patterns efficiently by prefix-projected pattern growth. *Proceedings 17th International Conference on Data Engineering*, pages 215–224, 2001. `doi:10.1109/ICDE.2001.914830`.

[87] Riccardo Pertile, Valeria Donisi, Laura Grigoletti, Andrea Angelozzi, Giuseppe Zamengo, Grazia Zulian, and Francesco Amaddeo. DRGs and other patient-, service-and area-level factors influencing length of stay in acute psychiatric wards: the veneto region experience. *Social psychiatry and psychiatric epidemiology*, 46(7):651–660, 2011. `doi:10.1007/s00127-010-0231-1`.

[88] Alistair EW Pollard, Tom J abd Johnson. The mimic-iii clinical database, 2016. `doi:10.13026/C2XW26`.

[89] Jennifer Pryor and John W. Chinneck. Faster integer-feasibility in mixed-integer linear programs by branching to force change. *Computers & Operations Research*, 38(8):1143 – 1152, 2011. URL: `http://www.sciencedirect.com/science/article/pii/S0305054810002546`, `doi:10.1016/j.cor.2010.10.025`.

[90] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. `doi:10.1023/A:1022643204877`.

[91] J. Ross Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., 1993. `doi:10.1007/BF00993309`.

[92] Alberto Rossi, Vera Morgan, Francesco Amaddeo, Marco Sandri, Michele Tansella, and Assen Jablensky. Psychiatric out-patients seen once only in south verona and western australia: a comparative case-register study. *Australian and New Zealand Journal of Psychiatry*, 39(5):414–422, 2005. `doi:10.1080/j.1440-1614.2005.01590.x`.

[93] Lucia Sacchi, Cristiana Larizza, Carlo Combi, and Riccardo Bellazzi. Data mining with temporal abstractions: learning rules from time series. *Data Min. Knowl. Discov.*, 15(2):217–247, 2007. `doi:10.1007/s10618-007-0077-7`.

[94] Pietro Sala. Approximate interval-based temporal dependencies: The complexity landscape. In *Temporal Representation and Reasoning (TIME), 2014 21st International Symposium on*, pages 69–78. IEEE, 2014. doi:10.1109/TIME.2014.20.

[95] Pietro Sala, Carlo Combi, Matteo Cuccato, Andrea Galvani, and Alberto Sabaini. A Framework for Mining Evolution Rules and Its Application to the Clinical Domain. In *2015 International Conference on Healthcare Informatics, ICHI 2015*, pages 293–302. IEEE Computer Society, 2015. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7349543, doi:10.1109/ICHI.2015.42.

[96] Pietro Sala, Carlo Combi, Matteo Mantovani, and Romeo Rizzi. Discovering evolving temporal information: Theory and application to clinical databases. *SN Computer Science*, 1(3):153, May 2020. doi:10.1007/s42979-020-00160-9.

[97] Ashok Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pages 432–444. Morgan Kaufmann, 1995. URL: http://www.vldb.org/conf/1995/P432.PDF.

[98] Yuval Shahar. A framework for knowledge-based temporal abstraction. *Artif. Intell.*, 90(1-2):79–133, 1997. doi:10.1016/S0004-3702(96)00025-2.

[99] Yuval Shahar. Dynamic temporal interpretation contexts for temporal abstraction. *Ann. Math. Artif. Intell.*, 22(1-2):159–192, 1998. doi:10.1023/A:1018998326167.

[100] Yuval Shahar. Knowledge-based temporal interpolation. *J. Exp. Theor. Artif. Intell.*, 11(1):123–144, 1999. doi:10.1080/095281399146643.

[101] Yuval Shahar and Carlo Combi. Temporal reasoning and temporal data maintenance in medicine: Issues and challenges. *Computers in Biology and Medicine*, 27(5):353–368, 1997. doi:10.1007/978-1-4419-6543-1.

[102] Yuval Shahar and Mark A. Musen. Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine*, 8(3):267–298, 1996. doi:10.1016/0933-3657(95)00036-4.

[103] Richard T Snodgrass, Michael H Böhlen, Christian S Jensen, and Andreas Steiner. Adding valid time to SQL/temporal. *ANSI X3H2-96-501r2, ISO/IEC JTC*, 1, 1996.

[104] Margarita Sordo, Gabriela Ochoa, and Shawn N. Murphy. A PSO/ACO approach to knowledge discovery in a pharmacovigilance context. In *GECCO (Companion)*, pages 2679–2684, 2009. doi:10.1145/1570256.1570382.

[105] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter M. G. Apers, Mokrane Bouzeghoub,

and Georges Gardarin, editors, *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 1996. doi:10.1007/BFb0014140.

[106] JE Tello, M Mazzi, M Tansella, P Bonizzato, J Jones, and F Amaddeo. Does socioeconomic status affect the use of community-based psychiatric services? a south verona case register study. *Acta Psychiatrica Scandinavica*, 112(3):215–223, 2005. doi:10.1111/j.1600-0447.2005.00558.x.

[107] Juan Eduardo Tello, Julia Jones, Paola Bonizzato, Mariangela Mazzi, Francesco Amaddeo, and Michele Tansella. A census-based socio-economic status (ses) index as a tool to examine the relationship between mental health services use and deprivation. *Social science & medicine*, 61(10):2096–2105, 2005. doi:10.1016/j.socscimed.2005.04.018.

[108] Hannu Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 134–145. Morgan Kaufmann, 1996. URL: http://www.vldb.org/conf/1996/P134.PDF.

[109] Natalia Vanetik, Ehud Gudes, and Solomon Eyal Shimony. Computing frequent graph patterns from semistructured data. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 458–465. IEEE Computer Society, 2002. URL: https://ieeexplore.ieee.org/xpl/conhome/8435/proceeding, doi:10.1109/ICDM.2002.1183988.

[110] Victor Vianu. Dynamic functional dependencies and database aging. *Journal of the ACM (JACM)*, 34(1):28–59, 1987. doi:10.1145/7531.7918.

[111] Jianyong Wang and Jiawei Han. BIDE: efficient mining of frequent closed sequences. In Z. Meral Özsoyoglu and Stanley B. Zdonik, editors, *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*, pages 79–90. IEEE Computer Society, 2004. URL: https://ieeexplore.ieee.org/xpl/conhome/9217/proceeding, doi:10.1109/ICDE.2004.1319986.

[112] Jianyong Wang and George Karypis. HARMONY: Efficiently mining the best rules for classification. In *Proceedings of SDM*, 2005. doi:10.1137/1.9781611972757.19.

[113] X Sean Wang, Claudio Bettini, Alexander Brodsky, and Sushil Jajodia. Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems (TODS)*, 22(2):115–170, 1997. doi:10.1145/249978.249979.

[114] Jef Wijsen. Reasoning about qualitative trends in databases. *Information Systems*, 23(7):463–487, 1998. doi:10.1016/S0306-4379(98)00023-4.

[115] Jef Wijsen. Temporal FDs on complex objects. *ACM Trans. Database Syst.*, 24(1):127–176, 1999. doi:10.1145/310701.310715.

[116] Jef Wijsen. Trends in databases: Reasoning and mining. *IEEE Trans. Knowl. Data Eng.*, 13(3):426–438, 2001. doi:10.1109/69.929900.

[117] Jef Wijsen. Temporal dependencies. In *Encyclopedia of Database Systems*, pages 2960–2966. Springer US, 2009. doi:10.1007/978-0-387-39940-9_396.

[118] Edi Winarko and John F. Roddick. ARMADA - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data Knowl. Eng.*, 63(1):76–90, 2007. doi:10.1016/j.datak.2006.10.009.

[119] World Health Organization and WHO Collaborating Centre for International Drug Monitoring. *The Importance of Pharmacovigilance.* Safety monitoring of medicinal products. World Health Organization, 2002. URL: https://apps.who.int/iris/handle/10665/42493/.

[120] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: a profile-based approach. In *Proceedings of SIGKDD*, 2005. doi:10.1145/1081870.1081907.

[121] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 721–724. IEEE Computer Society, 2002. URL: https://ieeexplore.ieee.org/xpl/conhome/8435/proceeding, doi:10.1109/ICDM.2002.1184038.

[122] Ying Yang, Geoffrey I. Webb, and Xindong Wu. Discretization methods. In Oded Maimon and Lior Rokach, editors, *The Data Mining and Knowledge Discovery Handbook.*, pages 113–130. Springer, 2005. doi:10.1007/0-387-25465-X_6.

[123] Osmar R Zaiane, Man Xin, and Jiawei Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries-ADL'98-*, pages 19–29. IEEE, 1998. doi:10.1109/ADL.1998.670376.

[124] Mohammed J. Zaki. Spade: an efficient algorithm for mining frequent sequences. In *Machine Learning Journal*, pages 31–60, 2001. doi:10.1023/A:1007652502315.

[125] Grazia Zulian, Valeria Donisi, Giacomo Secco, Riccardo Pertile, Michele Tansella, and Francesco Amaddeo. How are caseload and service utilisation of psychiatric services influenced by distance? a geographical approach to the study of community-based mental health services. *Social Psychiatry and Psychiatric Epidemiology*, 46(9):881–891, 2011. doi:10.1007/s00127-010-0257-4.

Chapter 8

List of Publications

In the following list, there is a summary of the publications or submissions written during this PhD. If the list of the authors is in alphabetical order, it symbolizes an equal contribution for that work.

- [30] Carlo Combi, Matteo Mantovani, and Pietro Sala. Discovering quantitative temporal functional dependencies on clinical data. In *2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, August 23-26, 2017*, pages 248–257. IEEE Computer Society, 2017. URL: https://ieeexplore.ieee.org/xpl/conhome/8030514/proceeding, doi:10.1109/ICHI.2017.80

- [73] Matteo Mantovani. Approximate temporal functional dependencies on clinical data. In *2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, August 23-26, 2017*, page 328. IEEE Computer Society, 2017. URL: https://ieeexplore.ieee.org/xpl/conhome/8030514/proceeding, doi:10.1109/ICHI.2017.30

- [74] Matteo Mantovani, Carlo Combi, and Milos Hauskrecht. Mining compact predictive pattern sets using classification model. In David Riaño, Szymon Wilk, and Annette ten Teije, editors, *Artificial Intelligence in Medicine - 17th Conference on Artificial Intelligence in Medicine, AIME 2019, Poznan, Poland, June 26-29, 2019, Proceedings*, volume 11526 of *Lecture Notes in Computer Science*, pages 386–396. Springer, 2019. doi:10.1007/978-3-030-21642-9_49

- [75] Matteo Mantovani, Carlo Combi, and Matteo Zeggiotti. Discovering and analyzing trend-event patterns on clinical data. In *2019 IEEE International Conference on Healthcare Informatics, ICHI 2019, Xi'an, China, June 10-13, 2019*, pages 1–10. IEEE, 2019. URL: https://ieeexplore.ieee.org/xpl/conhome/8895688/proceeding, doi:10.1109/ICHI.2019.8904774

- [96] Pietro Sala, Carlo Combi, Matteo Mantovani, and Romeo Rizzi. Discovering evolving temporal information: Theory and application to clinical databases. *SN Computer Science*, 1(3):153, May 2020. doi:10.1007/s42979-020-00160-9

Appendix A

The Computational Complexity of Checking *APE*-FDs

APE-FDs are proposed in Chapter 3. Here, we address the complexity of checking an *APE*-FD against an instance $\mathbf{r}$. We call this problem *Check-APE-FD*:

Problem 2. (Check-*APE*-FD). Given a temporal schema R, a *PE*-FD $[\Delta_k(\tau_J^R)]\ X\overline{Y} \to \overline{Z}$ on R, an instance $\mathbf{r}$ of R, and a real number $0 \leq \epsilon \leq 1$, determine whether $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \xrightarrow{\varepsilon} \overline{Z}$ or not.

Let us consider, for example, the *PE*-FD $[\Delta_{+\infty}(\tau_{PatId}^{ThCy})]Phys, \overline{Phys} \to \overline{Dos}$. We have proved above that $\mathbf{r} \not\models fd$. Figure 3-4 graphically reports all the possible $\tau_{PatId}^{\mathbf{r}'}$ where $\mathbf{r}'$ is obtained from $\mathbf{r}$ by deleting exactly one tuple. For example if $\mathbf{r}' = \mathbf{r} \setminus \{t_1\}$, it means that the dotted edge (t_1, t_2) has been removed. This means that t_1 and t_2 are not joined in $\tau_{PatId}^{\mathbf{r}'}$. Moreover if we take $\mathbf{r}' = \mathbf{r} \setminus \{t_2\}$ we have that both the edges (t_1, t_2) and (t_2, t_3) are removed and the dashed edge (t_1, t_3) turns out to be "active". This means that t_1 and t_2 are not joined in $\tau_{PatId}^{\mathbf{r}'}$ as well as t_2 and t_3, but t_1 and t_3 turn out to be joined in $\tau_{PatId}^{\mathbf{r}'}$ due to the absence of t_2. Let us observe that in this case the join operation involving t_1 and t_3 belongs to $\tau_{PatId}^{\mathbf{r}'}$ and not to $\tau_{PatId}^{\mathbf{r}}$. This specific behavior, in which the deletion of a tuple introduces additional, possibly different, constraints as a side effect, instead of just removing existing ones, gives us a hint on the problem *Check-APE-FD*. Such problem is not so easy to solve. Notice that $\mathbf{r} \setminus \{t_1\} \not\models [\Delta_{+\infty}(\tau_{PatId}^{ThCy})]Phys, \overline{Phys} \to \overline{Dos}$ as well, because of the pairs (t_2, t_3) and (t_6, t_7). However $\mathbf{r} \setminus \{t_2\} \models [\Delta_{+\infty}(\tau_{PatId}^{ThCy})]Phys, \overline{Phys} \to \overline{Dos}$ and thus we have that $\mathbf{r} \models [\Delta_{+\infty}(\tau_{PatId}^{ThCy})]Phys, \overline{Phys} \xrightarrow{\varepsilon} \overline{Dos}$ with $\epsilon = \frac{1}{8}$.

Therefore, problem *Check-APE-FD* belongs to the complexity class NP. In order to prove that, it suffices to apply a guess-and-check algorithm. First, this algorithm guesses a set $\mathbf{r}'$ with $|\mathbf{r}'| \leq \epsilon \cdot |\mathbf{r}|$. Then, if $\mathbf{r} \setminus \mathbf{r}' \models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$, the algorithm returns YES, otherwise *NO*. In the procedure above we implicitly make use of a function that verifies, given an instance $\mathbf{r}$ of R and a *PE*-FD $[\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$, whether $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$ holds or not. We can call this problem *Check-PE-FD*. Since there is no approximation, checking if $\mathbf{r} \models [\Delta_k(\tau_J^R)]X\overline{Y} \to \overline{Z}$ may be performed in polynomial time [32]. For this reason we can conclude that *Check-APE-FD* belongs to the complexity class NP. In the following, we

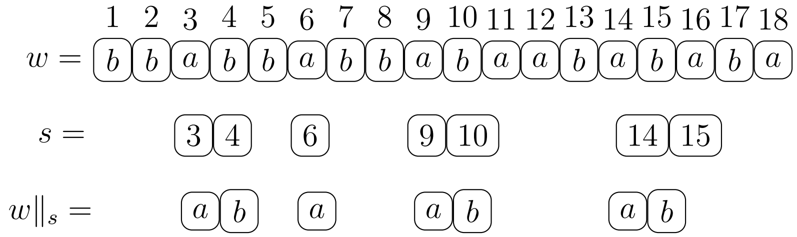


Figure A-1: An example of a word $w\|_s$ obtained by applying a sequence s to a word w.

will prove that Check-*APE*-FD is NP-hard even in the case of the most constrained kind of *APE*-FDs, which is represented by the class of simple update *APE*-FDs. From now on, we will consider Problem 2 only for simple update *APE*-FDs. Considering the inclusions shown in Figure 3-1, we can immediately conclude that our hardness result directly propagates to the other classes of *APE*-FDs.

In this section, we will make use of finite words w on a finite non-empty alphabet Σ (i.e., $w \in \Sigma^*$). We will use the standard notation $w[i]$ for denoting the i-th symbol of word w. Given a word w, we denote with $first(w)$ and $last(w)$ its first and its last element, respectively (i.e., $first(w) = w[1]$ and $last(w) = w[|w|]$). Moreover, a *finite increasing sequence of* $\mathbb{N}$ ($\mathbb{N}^>$-sequence) is a finite word s on $(\mathbb{N} \setminus \{0\})^*$, where for every i, i', with $1 \leq i < i' \leq |s|$, we have $s[i] < s[i']$[1]. Given a $\mathbb{N}^>$-sequence s we denote with $first(s)$ and $last(s)$ its first and its last element, respectively (i.e., $first(s) = s[1]$ and $last(s) = s[|s|]$). A $\mathbb{N}^>$-sequence s, for which for every i, with $1 \leq i < |s|$, we have $w[i+1] = w[i] + 1$, is called *strict* and we denote it with $[b, e]$, where $b = first(s)$ and $e = last(s)$. Given a word w and a $\mathbb{N}^>$-sequence s, we denote with $w\|_s$ the word $w\|_s = w[first(s)] \ldots w[last(s)]$ (for a graphical account of how a word is filtered by a sequence please refer to Figure A-1). Given a word w and a pair (b, e), with $1 \leq b \leq e \leq |w|$, we call the word $w\|_{[b,e]}$ a *slice* of w. Given two words $w_1, w_2 \in \Sigma^*$ we say that w_1 is a sub-sequence of w_2 (written $w_1 \sqsubseteq w_2$) if and only if there exists an $\mathbb{N}^>$-sequence s for which $w_1 = w_2\|_s$. For instance, $w_1 = abaabba$ is a sub-sequence of $w_2 = bbabbabbabaabababa$ with $s = 3\ 4\ 6\ 9\ 10\ 14\ 15$. A word w is *repetition free* if and only if for every $a \in \Sigma$ we have $|\{i : w[i] = a\}| \leq 1$. A word w is a *permutation* of Σ if and only if w is repetition free and $|w| = |\Sigma|$.

The proof that *Check-APE-FD* is NP-hard is done in two steps. First, we describe a known NP-Complete problem called *Common Permutation Problem* (*CP*-P for short). Then we introduce a problem called *Periodic Repair Problem* (*PR*-P) and we prove that *CP*-P may be reduced to it using logarithmic space. Finally we reduce the *PR*-P to Check-*APE*-FD using logarithmic space.

Let us begin with the Common Permutation Problem which has been proved to be NP-Complete in [40].

Problem 3. (*CP*-P). Given a finite alphabet Σ and two words w_1, w_2 over it, is there a permutation w_p of Σ for which $w_p \sqsubseteq w_1$ and $w_p \sqsubseteq w_2$?

[1] a $\mathbb{N}^>$-sequence is nothing more than a different representation for a finite set of positive naturals, but it turns out for our purposes to see it as a particular kind of word over positive naturals.

Consider $\Sigma = \{a, b, c\}$ we have that the pair $w_1 = bcbab$ and $w_2 = accaacb$ is a positive instance of Problem 3 because $cab \sqsubseteq w_1$ and $cab \sqsubseteq w_2$. On the other hand, the pair $w_1 = bcbac$ and $w_2 = acab$ is a negative instance of Problem 3 since both words do not share any permutation of Σ as their subsequence. More precisely, w_1 contains the permutations bca, bac and cba while w_2 contains the permutations acb and cab.

A word w is periodic if and only if for every pair of indexes (i, i'), with $1 \leq i, i' < |w|$, we have that $w[i] = w[i']$ implies $w[i + 1] = w[i' + 1]$. Let us observe that if w is repetition-free then it is periodic. Moreover, if w is periodic for every pair (b, e), with $1 \leq b \leq e \leq |w|$, we have that $w\|_{[b,e]}$ is periodic (i.e, every slice of a periodic word is itself periodic). The following lemma turns out to be useful for our reduction.

Lemma 1. Given a periodic word w, if w is not repetition-free then there exists an index $i < |w|$ such that $\mathit{last}(w) = w[i]$.

Proof. Since w is not repetition free, there exists two indexes i, i', with $1 \leq i < i' \leq |w|$, such that $w[i] = w[i']$. We prove the claim by induction on $\Delta = |w| - i'$. For the base of the induction we have $\Delta = 0$ and thus the claim trivially holds since i is the index we were looking for. Let us consider $\Delta = n + 1$. Since w is periodic and $w[i] = w[i']$ we have that $w[i + 1] = w[i' + 1]$. Thus positions $i + 1$ and $i' + 1$ witness a repetition and since $|w| - (i' + 1) < |w| - i' = \Delta$ we can apply the inductive hypothesis and prove our claim. □

In order to prove that *Check-APE-FD* is NP-Complete even for simple update *PE*-FD $[\Delta_k(\tau_J^R)]X \to \overline{X}$, we introduce the following intermediate problem called *Periodic Repair Problem* (*PR*-P for short):

Problem 4. (*PR*-P) Given a word $w = a_1 \dots a_n$, a finite alphabet Σ, and a natural number k, determine whether a periodic word $w' \sqsubseteq w$ exists such that $|w'| \geq k$.

Problem 4 belongs to the complexity class NP. A simple non-deterministic algorithm for *PR*-P guesses an $\mathbb{N}^>$-sequence s such that $|s| \geq k$ and $\mathit{last}(s) \leq |w|$ (i.e., s "chooses" only positions in $1 \dots |w|$). Then it suffices to check whether or not $w\|_s$ is periodic (periodicity checking may be performed in logarithmic space).

In the following, we describe how to reduce *CP*-P to *PR*-P. Let us consider two words w_1 and w_2 on an alphabet Σ with length n_1 and n_2, respectively. We assume without loss of generality that Σ is a finite subset of the negative integers (i.e., $\Sigma \subseteq \mathbb{Z}^-$). Let $n = \max(n_1, n_2)$ and $\sigma = |\Sigma|$. Let us consider the following word w over the alphabet $\Sigma \cup \{1, \dots, n\}$ ($\cdot$ is the classical word concatenation operator):

$$w = 1 \cdot \ldots \cdot n \cdot w_1 \cdot 1 \cdot \ldots \cdot n \cdot w_2 \cdot 1 \cdot \ldots \cdot n.$$

Finally, we put $k = 3n + 2\sigma$. Such reduction operates in logarithmic space. The following two lemmas prove the soundness and completeness of the above reduction.

Lemma 2. If there exists a permutation w_p of Σ which is a common subsequence of w_1, w_2, then there exists a $\mathbb{N}^>$-sequence s, with $|s| \geq 3n + 2\sigma$ and $\mathit{last}(s) \leq 3n + |w_1| + |w_2|$, such that $w\|_s$ is periodic.

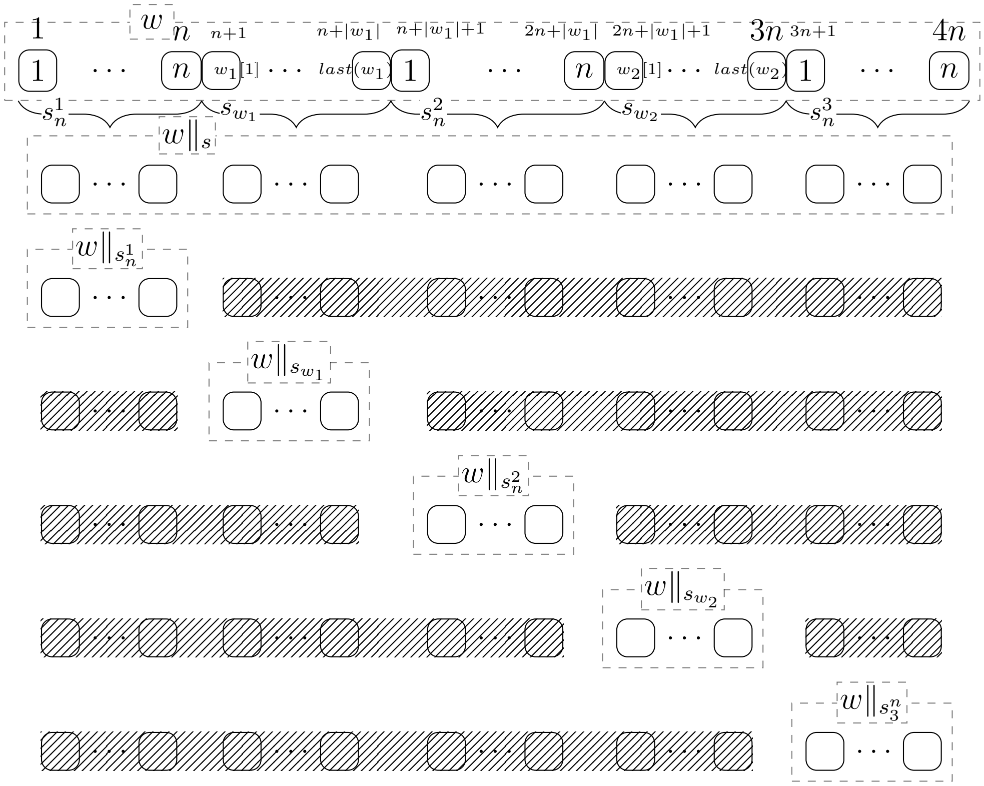


Figure A-2: A graphical account of how $s_n^1, s_{w_1}, s_n^2, s_{w_2}$, and s_n^3 filter blocks of w.

Proof. First, let us recall that w_p is a repetition-free sequence of symbols in $\mathbb{Z}^-$. By hypothesis we have $w_p \sqsubseteq w_1$ and $w_p \sqsubseteq w_2$ and thus there exists a pair of $\mathbb{N}^>$-sequence s_1 and s_2 with $|s_1| = \sigma$, $|s_2| = \sigma$ and $w_1\|_{s_1} = w_2\|_{s_2} = w_p$.

Let $\overline{s}_j$ with $j \in \{1, 2\}$ be the $\mathbb{N}^>$-sequence such that $|\overline{s}_j| = \sigma$ and for every $1 \le i \le \sigma$ we have $\overline{s}_j[i] = s_j[i] + nj + \sigma(j-1)$. Let us observe that $\overline{s}_j$ is a simple shift of the indexes in the sequence s_j with $j \in \{1, 2\}$. Then we may define s as follows:

$$s = \begin{array}{l} [1, n] \cdot \overline{s}_1 \cdot [n + |w_1| + 1, 2n + |w_1|] \cdot \overline{s}_2 \cdot \\ \cdot [2n + |w_1| + |w_2| + 1, 3n + |w_2| + |w_1|] \end{array}$$

By construction we have $w\|_s = 1 \dots n \cdot w_p \cdot 1 \dots n \cdot w_p \cdot 1 \dots n$. Since $\Sigma \cap \{1, \dots n\} = \emptyset$ there are not “conflicts” between the blocks $1 \dots n$ and w_p, thus we can conclude that $w\|_s$ is periodic. □

Lemma 3. If there exists a sequence s with $3n + 2\sigma \le |s| \le 3n + |w_1| + |w_2|$ for which $w\|_s$ is periodic, then there exists a permutation w_p of Σ which is a common subsequence of w_1, w_2.

Proof. First we define $s_n^1, s_{w_1}, s_n^2, s_{w_2}, s_n^3$ such that $s = s_n^1 \cdot s_{w_1} \cdot s_n^2 \cdot s_{w_2} \cdot s_n^3$ and $last(s_n^1) \le n$, $n + 1 \le s_{w_1}[1] \le last(s_{w_1}) \le n + |w_1|$, $n + |w_1| + 1 \le s_n^2[1] \le last(s_n^2) \le 2n + |w_1|$, $2n + |w_1| + 1 \le s_{w_2}[1] \le last(s_{w_2}) \le 2n + |w_1| + |w_2|$ and $n + |w_1| + |w_2| + 1 \le s_n^3[1]$.

Informally $s_n^1, s_{w_1}, s_n^2, s_{w_2}, s_n^3$ are the indexes in s that concern the sub-words $1 \dots n$ (first block), w_1, $1 \dots n$ (second block), w_2 and $1 \dots n$ (third block) respectively. A graphical account of this decomposition of $w\|_s$ is given in Figure A-2. This means that we may retrieve the sub-sequence selected by s on w restricted to the first block by means of the operation $w\|_{s_n^1}$. If we want to retrieve the sub-sequence selected by s on w restricted to the w_1 block we write $w\|_{s_{w_1}}$ and so on for the second block $1 \dots n$ (i.e., $w\|_{s_n^2}$), the block w_2 (i.e., $w\|_{s_{w_2}}$), and the third block $1 \dots n$ (i.e., $w\|_{s_n^3}$). Let us notice that $w\|_s$ is equal to $w\|_{s_n^1} \cdot w\|_{s_{w_1}} \cdot w\|_{s_n^2} \cdot w\|_{s_{w_2}} \cdot w\|_{s_n^3}$.

Suppose by contradiction that $|s_n^2| = 0$. Then we have that $w' = w\|_{s_{w_1}} \cdot w\|_{s_{w_2}}$ is a slice of $w\|_s$ and thus w' is periodic. Moreover we have that $w'' = w' \cdot (w\|_{s_n^3})$ is a slice of $w\|_s$ and thus w'' is periodic. Two cases may arise, i.e. either $|s_n^3| = 0$ or not.

If $|s_n^3| = 0$ we have that $w\|_s \sqsubseteq 1 \dots n \cdot w'$ and by a counting argument we have that $|w\|_s| \leq 3n$ which is a contradiction since $3n + 2\sigma \leq |w\|_s|$ and $\sigma > 0$ by definition.

If $|s_n^3| > 0$ we prove that w'' is repetition free. Again by contradiction, from Lemma 1 we will have that, since w'' is periodic, there must exists an index $i < |w''|$ for which $w''[i] = w''[n]$. However $|s_n^3| > 0$ implies $w''[n] \in \{1, \dots, n\}$ and thus, since $\Sigma \cap \{1, \dots, n\} = \emptyset$, such i' cannot exist (contradiction). If w'' is repetition free we have that $|w''| \leq \sigma + n$ and since $w\|_s \sqsubseteq 1 \dots n \cdot w''$ we have that $|w\|_s| \leq 2n + \sigma$, which contradicts $|w\|_s| \geq 3n + 2\sigma$.

We have now that $|s_n^2| > 0$. Consider the slice $w' = w\|_{s_{w_1}} \cdot w\|_{s_n^2}$: we have just proved that $|w\|_{s_n^2}| > 0$ and we have that w' is periodic being a slice of $w\|_s$. By applying Lemma 1 as we did above we can claim that $w\|_{s_{w_1}}$ is repetition free and thus $|w\|_{s_{w_1}}| \leq \sigma$. Suppose now by contradiction that $w\|_{s_{w_2}}$ is not repetition free. If $|s_n^3| > 0$ we reach immediately a contradiction by applying Lemma 1 on the word $w\|_{s_{w_2}} \cdot w\|_{s_n^3}$. Then we have $|s_n^3| = 0$ and by definition $|w\|_{s_{w_2}}| \leq n$. This implies $w\|_s \sqsubseteq 1 \dots n \cdot w\|_{s_{w_1}} \cdot 1 \dots n \cdot w\|_{s_{w_2}}$ which means $|w\|_s| \leq 3n + \sigma$ (contradiction).

At this point we have that both $w\|_{s_{w_1}}$ and $w\|_{s_{w_2}}$ are repetition free and thus $|w\|_{s_{w_1}}| \leq \sigma$ and $|w\|_{s_{w_2}}| \leq \sigma$. Since $3n + 2\sigma \leq |w\|_s|$ we have that $|w\|_{s_{w_1}}| = \sigma$ and $|w\|_{s_{w_2}}| = \sigma$ and thus both $w\|_{s_{w_1}}$ and $w\|_{s_{w_2}}$ are permutations of Σ. It remains to prove that they are the same permutation. Let us observe that, since $|w\|_{s_{w_1}}| = |w\|_{s_{w_2}}| = \sigma$ and $|w\|_s| \geq 3n + 2\sigma$, by a counting argument we have that $w\|_{s_n^1} = w\|_{s_n^2} = w\|_{s_n^3} = 1 \dots n$. and thus $w\|_s = 1 \dots n \cdot w\|_{s_{w_1}} \cdot 1 \dots n \cdot w\|_{s_n^2} \cdot 1 \dots n$.

Suppose by contradiction that there exists i, with $1 \leq i \leq \sigma$, such that $w\|_{s_{w_1}}[i] \neq w\|_{s_{w_2}}[i]$, and let i be the minimum index that fulfills such a property. Two cases may arise:

1. If $i = 1$ we have that $w\|_s[n+1] = w\|_{s_{w_1}}[i]$ and $w\|_s[2n + \sigma + 1] = w\|_{s_{w_2}}[i]$. Let us recall that $w\|_s[n] = w\|_s[2n + \sigma] = n$ and thus by periodicity of $w\|_s$ we have $w\|_s[n+1] = w\|_s[2n + \sigma + 1]$ (contradiction).

2. If $i > 1$ since $w\|_s$ is periodic we have that $w\|_{s_{w_1}}[i-1] \neq w\|_{s_{w_2}}[i-1]$ but this contradicts the minimality in the choice of i.

□

Now we reduce Problem PR-P to *Check-APE-FD* in logarithmic space. Suppose that we have an instance of PR-P consisting of a word $w \in \Sigma^*$ and a natural number k. We define the instance $\mathbf{r}_w$ on the temporal schema $R = \{J, X\} \cup VT$ as $\mathbf{r}_w = \{t \mid t[J] = 0 \wedge \exists i(t[X] = w[i] \wedge t[VT] = i)\}$ and we put $\epsilon_{w,k} = \frac{|w|-k}{|w|}$. The pair w, k is a positive instance of PR-P if and only if the triple $[\Delta_{+\infty}(\tau_J^R)]X \rightarrow \overline{X}$, $\mathbf{r}_w$ and $\epsilon_{w,k}$ is a positive instance of *Check-APE-FD*. $[\Delta_{+\infty}(\tau_J^R)]X \rightarrow \overline{X}$, $\mathbf{r}_w$ and $\epsilon_{w,k}$ may be built using logarithmic space on the input w, k. Finally, we can conclude this section by explicitly providing the desired result.

Theorem 5. Problem Check-APE-FD is NP-Complete.