Knowledge and Information Systems (2023) 65:4797–4846
https://doi.org/10.1007/s10115-023-01910-w

REGULAR PAPER

# Continuous prediction of a time intervals-related pattern's completion

**Nevo Itzhak[1] · Szymon Jaroszewicz[2,3] · Robert Moskovitch[1]**

Received: 2 December 2022 / Revised: 27 April 2023 / Accepted: 11 May 2023 /
Published online: 24 June 2023


**Abstract**

In many daily applications, such as meteorology or patient data, the starting and ending times of the events are stored in a database, resulting in time interval data. Discovering patterns from time interval data can reveal informative patterns, in which the time intervals are related by temporal relations, such as *before* or *overlaps*. When multiple temporal variables are sampled in a variety of forms, and frequencies, as well as irregular events that may or may not have a duration, time intervals patterns can be a powerful way to discover temporal knowledge, since these temporal variables can be transformed into a uniform format of time intervals. Predicting the completion of such patterns can be used when the pattern ends with an event of interest, such as the recovery of a patient, or an undesirable event, such as a medical complication. In recent years, an increasing number of studies have been published on time intervals-related patterns (TIRPs), their discovery, and their use as features for classification. However, as far as we know, no study has investigated the prediction of the completion of a TIRP. The main challenge in performing such a completion prediction occurs when the time intervals are coinciding and not finished yet which introduces uncertainty in the evolving temporal relations, and thus on the TIRP's evolution process. To overcome this challenge, we propose a new structure to represent the TIRP's evolution process and calculate the TIRP's completion probabilities over time. We introduce two continuous prediction models (CPMs), segmented continuous prediction model (SCPM), and fully continuous prediction model (FCPM) to estimate the TIRP's completion probability. With the SCPM, the TIRP's completion probability changes only at the TIRP's time intervals' starting or ending point. The FCPM incorporates, in addition, the duration between the TIRP's time intervals' starting and

✉ Nevo Itzhak
nevoit@post.bgu.ac.il

Szymon Jaroszewicz
s.jaroszewicz@ipipan.waw.pl

Robert Moskovitch
robertmo@bgu.ac.il

1 Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

2 Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

3 Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland

ending time points. A rigorous evaluation of four real-life medical and non-medical datasets was performed. The FCPM outperformed the SCPM and the baseline models (random forest, artificial neural network, and recurrent neural network) for all datasets. However, there is a trade-off between the prediction performance and their earliness since the new TIRP's time intervals' starting and ending time points are revealed over time, which increases the CPM's prediction performance.

**Keywords** Time interval data · Time interval patterns · Continuous prediction

## 1 Introduction

Using frequent temporal patterns, such as sequential patterns, time intervals-based patterns, or time series trends, whether provided by a domain expert or discovered by a mining process [1–5], were used for temporal knowledge discovery [6], clustering [7], or as features for classification [8–11] or prediction of certain outcomes [12, 13]. Estimating the completion probability of a temporal pattern of interest can be used for predicting the future state of ongoing cases in a process [14–16] or when a temporal pattern ends with an event of interest, such as the recovery of a patient, or an undesirable event, such as death or a medical complication. In medicine, for example, it can be used to continuously predict a clinical outcome of a monitored patient in an intensive care unit (ICU) [12, 17–19], where the clinical team needs to be warned of potential complications to enable early intervention and ideally, prevention. For example, an acute blood pressure elevation [12] was one of the undesirable events used in this study, in which the data were extracted from real ICU patients. Additionally, in predictive maintenance, it is desirable to continuously monitor essential machinery's condition to track performance and detect possible defects that could result in a system crash.

The idea of predicting the temporal patterns' completion was applied with sequential patterns in several domains, such as complex activity recognition [8, 20, 21] and recommendations [22, 23]. However, many events in daily applications, such as meteorology data, stock fluctuations, or patient data, are not instantaneous events. In some domains, events may last along a certain time period, in which the starting and ending times are known and stored in a database, which results in symbolic time interval data. Discovering patterns while considering the starting and ending times of events can reveal more informative patterns. For example, a pattern from time interval data may be that hospitalized patients with COVID-19 frequently start with symptoms of "fever" and "cough," and a week or so later also begin experiencing shortness of breath [24], in which the symptoms were not ended until the ICU admission time.

In previous studies, models based on time intervals patterns [12, 13, 19] or symbolic time points [25] achieved better classification performance than using deep neural networks on the raw temporal data. An explanation for that could be due to the neural networks' limitations as they require a large amount of data, and sparsity occurs when handling heterogeneous multivariate temporal data with different granularity. In this paper, we investigate the usage of time interval patterns in a novel manner that differs from the approaches found in the existing literature. The main innovation of this paper is the introduction of the problem, challenges, and our proposed methodology for a continuous time intervals-related pattern completion estimation, which, as far as we know, no previous study has investigated. The paper presents a methodology for continuously predicting the completion of a time intervals-

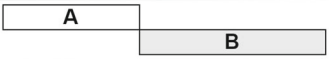
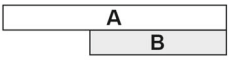
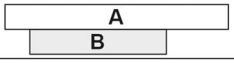
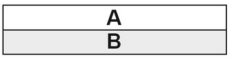
| Relation | | Schematic Representation | Endpoints Representation |
|---|---|---|---|
| A before B | < |  | A+ < A- < B+ < B- |
| A meets B | *m* |  | A+ < A- = B+ < B- |
| A overlaps B | *o* |  | A+ < B+ < A- < B- |
| A finished-by B | *fi* |  | A+ < B+ < A- = B- |
| A contains B | *c* |  | A+ < B+ < B- < A- |
| A starts B | *s* |  | A+ = B+ < A- < B- |
| A equals B | = |  | A+ = B+ < A- = B- |

**Fig. 1** Schematic and endpoints representation of Allen's seven temporal relations between a pair of STIs (*A* and *B*, in this figure)

related pattern, based on partial information, which can be applied to various domains and applications such as predictive process monitoring or clinical event prediction.

A *symbolic time interval (STI)* consists of starting and ending points and a symbol from an ordered alphabet. For simplicity, the denotation of an STI will be identical to its symbol throughout the paper. Additionally, $A+$ and $A-$ are used when the starting and ending point of STI $A$ are referred to (formal definitions are introduced in Sect. 2). To represent the temporal relation between each pair of STIs, typically, Allen's temporal intervals algebra [26] is used (see Fig. 1). A *time intervals-related pattern* (TIRP) is defined by a set of ordered STIs and the conjunction of all the temporal relations between each pair of those STIs [2–4, 27]. For example, the TIRP illustrated in Fig. 2 is defined by four STIs and six temporal relations (Fig. 2.ii). The durations of the TIRP STIs are not part of their definition, which enables the discovery and detection of frequent TIRPs. Since the TIRP's STIs' duration may vary from different pattern's instances, the schematic representation (Fig. 2.i) presents a possible instance of the pattern. The task of continuous estimation of a TIRP's completion is challenging, mainly due to the variability in the STIs' duration and gaps between the STIs of the pattern's instances, and in this study, we introduce and propose solutions. However, the temporal variables may be of heterogeneous nature, such as in medicine, which might contain lab test results [18], drug exposure periods, or the body's vital signs [12]. As explained in the next paragraph, time interval mining methods are suitable for analyzing heterogeneous multivariate data that were transformed into a uniform representation of STIs.

The growing collection and availability of time-stamped electronic data in various domains, such as transportation [28], medicine [29], and security [30], provides exceptional opportunities to discover new actionable temporal knowledge from multivariate, time-oriented data. As shown in Fig. 3, the temporal variables could be sampled at a fixed frequency (Fig. 3.FF), obtained by irregular sampling (Fig. 3.IS), represented by instantaneous events (Fig. 3.IE) or events with duration (Fig. 3.STI). Traditional time series analysis methods cannot often incorporate such heterogeneous longitudinal data and expect all the variables to be sampled in the same fixed frequency. Other methods, such as sequential mining, can

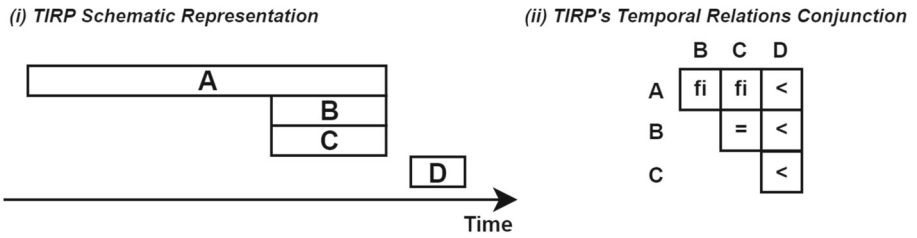



**Fig. 2** A schematic representation (i) and temporal relations conjunction (ii) of a TIRP that has four STIs and six Allen's temporal relations

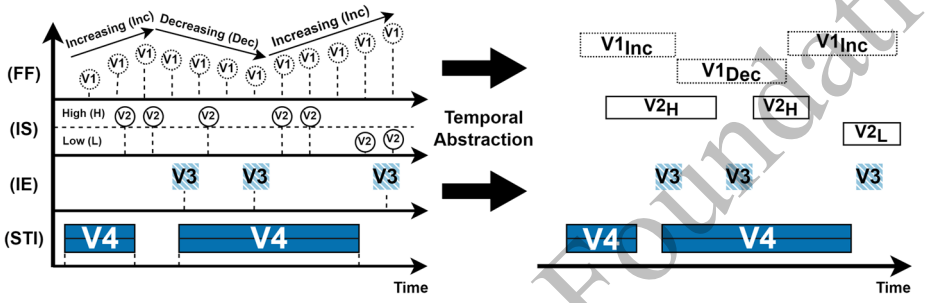



**Fig. 3** An entity described by heterogeneous multivariate temporal data, containing: fixed frequency (FF) sampling, irregular sampling (IS), instantaneous events (IE), and raw STIs. The temporal abstraction process transforms a series of raw time-stamped data points into a uniform representation of the STIs series

analyze only instantaneous events (Fig. 3.IE), but not events with a duration (Fig. 3.STI). For that, *temporal abstraction* [9, 31, 32] is increasingly being used to transform various temporal variables into a uniform representation of STIs. Thus, using TIRPs and continuously estimating their completion can be useful in any temporal variables' forms, which makes it widely applicable in various real-life data. For example, in the early prediction of an outcome [12, 18], monitoring temporal progress [14], or detecting temporal abnormalities in a real-time fashion. Even though temporal abstraction allows considerations of heterogeneous data in a unified way, it is not necessarily part of the prediction process while using TIRPs, since the data can be composed of only raw STIs.

Temporal abstraction is typically performed in two ways: *state abstraction*, in which the values are classified into states, based on given cutoffs, and later concatenated into STIs, or *gradient abstraction*, in which they are segmented into increasing or decreasing periods according to the first derivative. For example, in Fig. 3, variable *V1* is abstracted into increasing (*Inc*) and decreasing (*Dec*) STIs using gradient abstraction, while variable *V2* is abstracted into STIs using state abstraction, based on a single cutoff, into two states: high (*H*) or low (*L*). Once a database of STIs is created, whether abstracted (Fig. 3.V1 and 3.V2) or raw (Fig. 3.V3 and 3.V4), time intervals analysis can be performed, such as TIRPs similarity [33], a discovery of frequent TIRPs [2–5, 27, 34, 35], or a continuous prediction of a TIRP's completion.

However, while continuously predicting the TIRP's completion, a noticeable problem arises when the STIs are unfinished and coinciding. This results in uncertainty about the actual evolving temporal relation between each pair of unfinished coinciding STIs (Definition 5). Moreover, the representation of transition probabilities among STIs becomes, in such cases, challenging. In this paper, we propose a new representation to overcome those challenges

that are unique for TIRPs since they include STIs that may have various potential pairwise temporal relations. In contrast, those challenges do not exist in sequential patterns, in which the events are instantaneous, and the temporal relations are only *before* or *co-occur* with other events. This sequential pattern's property makes it easier to define the pattern's evolution process since there is no uncertainty about temporal relations between its instantaneous events.

This paper focuses on predicting the completion probability for a single TIRP, which proved to be a sufficiently challenging problem. The models were evaluated separately on TIRPs, ending with a target event of interest. In the future, we intend to expand the proposed methods to continuously consider multiple patterns. Considering multiple patterns can be used for continuous prediction of real-life tasks such as outcome prediction of common complications in critically ill patients [12, 18].
A model based on multiple patterns that end with the event of interest is expected to be more accurate, have better data coverage, and improve generalization over models based on a single pattern.
The main contributions of the paper are:

1. Defining the problem of the continuous prediction of a TIRP's completion and noting the unfinished coinciding STIs challenge. Then, we propose a representation to overcome this challenge.
2. Introducing novel methods for continuous prediction of a TIRP's completion and evaluating them on real-life medical and non-medical datasets.

The rest of this paper is organized as follows: we start by reviewing the literature (Sect. 2), considering topics of temporal abstraction and relations, time interval data analysis and their applications, and sequential patterns' completion. We then proceed with the Problem Statement and Methods (Sects. 3 and 4), Evaluation and Results (Sects. 5 and 6), and finally summarize the work, present conclusions, and discuss future work (Sect. 7).

## 2 Background

### 2.1 Temporal abstraction

*Temporal abstraction* refers to the process of transforming continuous raw heterogeneous temporal data into a series of STIs (see Definition 1). Such a representation provides a uniform format for the heterogeneous temporal variables (e.g., time series [36, 37] or instantaneous events), enabling the analysis of their temporal relations.

**Definition 1** A *symbolic time interval (STI)* $I = (s, e, sym)$, is a triplet of start-time $s \in \mathbb{R}_{\geq 0}$, end-time $e \in \mathbb{R}_{\geq 0}$, $e \geq s$ and a symbol $sym$ ($sym \in \Sigma$) from an ordered alphabet $\Sigma$.

For simplicity, the denotation of an STI will be identical to its symbol throughout the paper. Additionally, $I_s$ and $I_e$ are used when the start-time and end-time of STI $I$ are referred to.

**Definition 2** A *time interval endpoint (tiep)* is a triplet $(t, type, sym)$ consisting of a time stamp $t \in \mathbb{R}_{\geq 0}$, an endpoint type, which can be either starting (+) or ending (−), and a symbol ($sym \in \Sigma$) from an ordered alphabet $\Sigma$.

The total order of a set of *tieps* (see Definition 2) is defined based on their time stamps, which are real numbers. For an STI $A = (A_s, A_e, \text{"}A\text{"})$, the starting and ending *tieps* are

defined, respectively, as $A+ = (A_s, +, \text{“}A\text{”})$ and $A- = (A_e, -, \text{“}A\text{”})$. Based on this notation and the total order between real numbers, the *tieps* can be used in inequalities defining temporal relations, while their structure will be exploited in algorithms in the following sections.

Typically, there are two temporal abstraction approaches, gradient, and state abstraction, as explained in the Introduction (Sect. 1) and illustrated in Fig. 3. The gradient abstraction determines the change direction of the variable values, such as increasing heart rate values.

For example, in Fig. 3, variable *V1* is abstracted into an interval-based representation using gradient abstraction based on the first derivative having two values: increasing (*Inc*) and decreasing (*Dec*).

When state abstraction is used, the cutoffs can be given by a domain expert or acquired in a data-driven fashion. For example, in Fig. 3, variable *V2* is abstracted into an interval-based representation using state abstraction, based on a single cutoff, into two states: high (*H*) and low (*L*). A few data-driven methods exist to determine these cutoffs [38], but we list those commonly used for time-based data. The most basic one is the equal width discretization (EWD) that divides the range of values uniformly into bins with the same value range. Equal frequency discretization (EFD) aims at having bins leading to uniform distribution of values in bins, or equivalently, equal probabilities of ending up in each bin; the cutoffs are determined accordingly. A popular method for time series discretization in the data mining community is symbolic aggregate approximation (SAX) [39]. SAX reduces the temporal dimensionality and transforms the time series into symbolic strings assuming that the processed data follows the Gaussian distribution.

An important part of the temporal abstraction process is choosing a suitable granularity of representation for the raw data, which should reduce noise, economize storage, and speed up temporal data processing. A well-defined procedure, named piecewise aggregate approximation (PAA) [40, 41], reduces dimensions longitudinally (increasing the temporal granularity) and is used in this paper. It is based on dividing the temporal data into equal duration segments, represented by each segment's mean values.

### 2.2 Temporal relations

Allen [26] defined seven temporal relations (and their inverses) between a pair of STIs. The seven temporal relations: *before* ($<$), *meets* ($m$), *overlaps* ($o$), *starts* ($s$), *finished-by* ($fi$), *equals* ($=$), and *contains* ($c$) are shown in Fig. 1. Let us now define the lexicographical order between STIs.

**Definition 3** A lexicographical STI series $IS$ is an STI series, sorted in the lexicographical order of the starting *tiep*, ending *tiep*, and symbol (sym), i.e., $IS = \{I^1, ..., I^k\}$, such that: $\forall I^i, I^j \in IS\ (i < j) \wedge [(I^i+ < I^j+) \vee ((I^i+ = I^j+) \wedge (I^i- < I^j-)) \vee ((I^i+ = I^j+) \wedge (I^i- = I^j-) \wedge (I^i_{sym} < I^j_{sym}))]$.

In this study, the STIs will always be ordered lexicographically. Thus, the seven temporal relations can be used without their corresponding inverse relations. Additionally, in a specific STI series, the exact events cannot co-occur, and as a result, STIs with the same symbols cannot overlap.

Allen's temporal relations are based on the assumption that an event has a starting and an ending point. Freska [42] has modified Allen's interval-based representation of temporal relations such that they can be used, rather naturally, for reasoning with incomplete knowledge. Relations among semi-intervals rather than intervals are used as the basic knowledge

units. Semi-intervals correspond to temporal beginnings or endings of events. In this study, Allen's seven temporal relations are used; however, the challenges with incomplete knowledge discussed in Freska's study [42] were considered to address uncertainties during the continuous prediction.

### 2.3 Frequent TIRP discovery

Once the data are transformed into a series of STIs, time intervals-related patterns can be discovered. We refer to these discovered STIs-based patterns as TIRPs and define them formally in Definition 4.

**Definition 4** A non-ambiguous lexicographic time intervals-related pattern (TIRP) $Q$ is defined as a pair $Q = (IS, R)$, where $IS = \{I^1, \ldots, I^k\}$ is a lexicographical STI series (Definition 3) of $k$ STIs and $R = \{r(I^i, I^j) : 1 \leq i < j \leq k\}$ is a set that defines all the temporal relations between each of the $(k^2 - k)/2$ pairs[1] of STIs in $IS$.

An example schematic representation and the corresponding conjunction of temporal relations of a TIRP are presented in Fig. 2. The conjunction of the temporal relations, typically presented using the half matrix representation (Fig. 2.ii), constitutes a canonical, non-ambiguous representation of pairwise temporal relations among the STIs of a lexicographically ordered TIRP. The half matrix representation (as opposed to the full matrix) is possible because each of Allen's seven temporal relations has an inverse. The canonical aspect is due to the lexicographical ordering leading to a unique half matrix for each TIRP. In contrast, the schematic representation (Fig. 2.i) presents one of many possible instances of the pattern. The STIs' durations are not part of a pattern definition, and other instances of the same pattern might have different durations of STIs.

A pattern is called frequent if its vertical support exceeds a pre-defined minimum threshold. Given a database DB of $|DB|$ unique entities, the *vertical support* $VS(DB, Q)$ of a TIRP $Q$ is denoted by the cardinality of the set $DB^Q$ of distinct entities within which $Q$ holds at least once, divided by the total number of entities $|DB|$, $VS(DB, Q) = |DB^Q|/|DB|$.

Several TIRP mining methods have been developed in the past two decades [4, 5, 27, 34, 35], most of which use Allen's definition to represent relations between pairs of STIs, which are used to define a TIRP. Hoppner [2] was the first to define non-ambiguous TIRPs (Definition 4) using a set of STIs and the conjunction of the temporal relations between each pair of STIs. Papapetrou [3] proposed a hybrid approach (H-DFS), which first indexes the pairs of time intervals and extends frequent TIRPs. Moskovitch and Shahar [27] introduced the KarmaLego algorithm that exploits the transitivity of temporal relations [26] to generate candidates efficiently and completely, which we use in this study. More details about the evolution of time interval mining can be found in [5, 34, 35].

### 2.4 Applications of frequent TIRPs

Frequent TIRPs are typically used as features for multivariate time series classification or prediction, as proposed first by Patel et al. [10]. The approach was inspired by the bag-of-words representation in text categorization [43], in which words are used as features for the classification of a given document. Batal et al. [11] proposed classifying electronic health

[1] The formula $(k^2 - k)/2$ follows from the binomial coefficient $\binom{k}{2} = k(k-1)/2 = (k^2 - k)/2$, where 2 stands for pairs of temporal relations and $k$ is the number of STIs.

records (EHR) data using a small set of predictive and non-spurious TIRPs. The Maitreya framework for outcomes prediction in EHR data based on frequent TIRPs was suggested by Moskovitch et al. [44]. The framework learns frequent TIRPs only from patients having the outcome, using the KarmaLego algorithm. Then, the patients' TIRPs are detected in both classes, and a classifier is induced. The framework was recently extended to predict acute hypertensive episodes in ICU patients [12] and the first fall in older care home residents [45].

The extracted TIRPs were used independently as features for the classifier in the aforementioned studies while ignoring the longitudinal order within those TIRPs. To address this challenge, Novitski et al. [13] proposed to feed sequential neural networks with the extracted frequent TIRPs. This approach was reported to be more accurate than using sequential neural networks trained on raw data for predicting mortality in elderly patients with diabetes. Liu et al. [21] suggested a semantic-based probabilistic framework for STI data that can be used to answer varied semantic-level queries in a unified way, such as predicting future activities given observed ones. However, their paper focused on classification in the domain of human activity recognition while ignoring the unfinished coinciding STIs challenge, which is necessary to continuously predict STI data.

To the best of our knowledge, no previous study has investigated the task of continuous prediction of a TIRP's completion.

### 2.5 Sequential patterns' completion

Prediction of sequential patterns' completion is useful in many tasks, such as complex activity recognition [8, 20, 21] and recommendations [22, 23]. In code recommendations, it was suggested to use sequential patterns to mine coding patterns from the project repository, and once the developer's code coincides with a beginning of a sequential pattern, the pattern is suggested [23]. Zhu et al. [22] introduced a recommendation system for travel products that is based on frequent sequential patterns, in which the sequential patterns comprised of the visited web pages' semantic descriptions and their target products. A model based on frequent sequential patterns is trained to recommend the highest-scored target product given a user's click-stream.

In sequential patterns, the temporal relation between each pair of a pattern's items is restricted to an item being *before* or *co-occurring* with other items. This property makes it easier to define the pattern's evolution process, avoiding the uncertainty about temporal relations between items. In contrast, when continuously predicting a TIRP's completion in STI data, a noticeable problem arises when the STIs are coinciding and unfinished.

Meaningful contributions have been made in the past two decades in analyzing STI data [2–5, 27] and its use in multivariate temporal data analysis after employing temporal abstraction [10, 12, 46]. However, continuous prediction of a TIRPs' completion, which can be used for ongoing events prediction, was not studied yet. Our study proposes a novel approach for a continuous TIRP's completion prediction in heterogeneous multivariate temporal data, which addresses the unfinished coinciding STIs challenge and considers the durations between the TIRP's STIs' *tieps* to allow for fully continuous prediction.

## 3 Problem statement

Forasmuch as no previous study has investigated the task of continuous prediction of a TIRP's completion, this section is devoted to describing and formalizing the task (Sect. 3.1) and introducing its fundamental challenges (Sects. 3.2–3.3).

Given raw heterogeneous multivariate temporal data, the data are first converted into an STI format using temporal abstraction methods (Sect. 2.1). The STIs are used as a uniform format for analyzing the temporal relations among them (Sect. 2.2). Then, a TIRP of interest can be discovered from the STI data using time intervals mining methods (Sect. 2.3) or can be provided by domain experts.

### 3.1 Problem formulation

A model $M$ estimates the probability of observing the remaining part of a TIRP $Q$, given the observed part of $Q$ at $t_c$, which will be referred to as the TIRP $Q$'s completion. Given a TIRP $Q$ and a database $DB$, a continuous prediction model $M$ that estimates the TIRP $Q$'s completion probability is created. An estimation is provided at each *current* time point $t_c$, and changes as a given instance of $Q$ evolves over time. The database $DB$ comprises $|DB|$ entities (e.g., patients), where each entity contains a lexicographically ordered STI series (Definition 3).

We now define several terms which are used to describe our models. Let $p_{t_c}$ denote a prefix representing the observed part of $Q$ at $t_c$, while $s_{t_c}$ denote a suffix representing the remaining part of $Q$ at $t_c$ that is expected to occur. Note that the TIRP's prefix notion will be refined in Sect. 4.1 after presenting the task challenges in this section. Thus, to estimate the TIRP $Q$'s completion probability $Pr(Q \mid t_c)$, at time point $t_c$, the following simplistic model (Formulas 1, 2, and 3) can be used

$$Pr(Q \mid t_c) = Pr(s_{t_c} \mid p_{t_c}) \tag{1}$$

$$= \frac{Pr(p_{t_c}, s_{t_c})}{Pr(p_{t_c})} \tag{2}$$

$$= \frac{Pr(Q)}{Pr(p_{t_c})}. \tag{3}$$

This simplistic formula typically represents the confidence of a rule in sequential patterns. However, there is no uncertainty about temporal relations between its instantaneous events in sequential patterns, making the computation easier. In contrast, continuously predicting the TIRP's completion results in uncertainty about the actual evolving temporal relation between each pair of unfinished coinciding STIs, as we demonstrate in this section.

Formula 1 follows from the fact that the probability $Pr(Q \mid t_c)$ is, by definition, the probability that the suffix $s_{t_c}$ occurs, given that, at time $t_c$, the prefix $p_{t_c}$ has already been observed. Formula 2 follows the definition of conditional probability. The denominator $Pr(p_{t_c})$ is the probability of the prefix $p_{t_c}$ to occur, while the numerator $Pr(p_{t_c}, s_{t_c})$ represents the probability of the prefix $p_{t_c}$ to occur and to be *followed* by the suffix $s_{t_c}$. Lastly, the equality in Formula 3 results from the fact that the occurrence of $p_{t_c}$ followed by $s_{t_c}$ is equivalent to the occurrence of the TIRP $Q$.

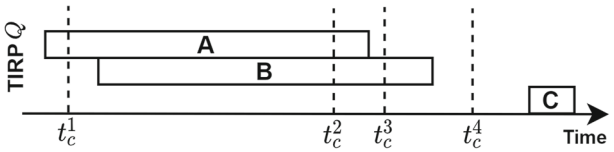



**Fig. 4** A TIRP $Q$ has three STIs: $A$, $B$, and $C$, in which $Q = \{A\ overlaps\ B, A\ before\ C, B\ before\ C\}$. At any time point (e.g., $t_c^1, t_c^2, t_c^3, t_c^4$), we aim to estimate the probability of the TIRP $Q$'s completion

### 3.2 An overview of model estimation procedure

In this subsection, we illustrate the calculations required for computing the TIRP's completion probabilities on a simple example highlighting the problems which arise during such computations.

To estimate the completion probability of a TIRP $Q$, we use Formula 3 ($Pr(Q)/Pr(p_{t_c})$), and to apply it continuously, we need to count the number of times each $p_{t_c}$ of $Q$ occurs in the database, as well as the number of times $p_{t_c}$ is followed by $s_{t_c}$ (i.e., $Q$ completes). This calculation answers the following question: "Out of all the times we saw $p_{t_c}$, how many times was it followed by $s_{t_c}$ (i.e., $Q$ has unfolded to completion)?" Since the database $DB$ comprises multiple entities (e.g., patients), and each entity contains a lexicographically ordered STI series, instances of $Q$ and $p_{t_c}$ may be discovered more than once in a single entity. Each such instance is *counted separately* in the computation.

Applying Formula 3 to a relatively simple example sheds light on the challenges that arise while continuously predicting TIRP's completion. In Fig. 4, a TIRP $Q = \{A\ overlaps\ B, A\ before\ C, B\ before\ C\}$ is shown, together with four time points $t_c^1, t_c^2, t_c^3, t_c^4$ chosen to demonstrate various types of challenges.

At time point $t_c^4$, the observed part of $Q$ is $p_{t_c^4} = \{A\ overlaps\ B\}$, and the remaining part of $Q$, that is expected to occur, since STI $C$ was not observed, is $s_{t_c^4} = \{A\ before\ C, B\ before\ C\}$. Formula 4 shows the TIRP $Q$'s completion probability $Pr(Q \mid t_c^4)$ at time $t_c^4$ calculated based on Formula 3

$$Pr(Q \mid t_c^4) = \frac{Pr(Q)}{Pr(A\ o\ B)}. \tag{4}$$

In Formula 4, the numerator *Pr(Q)* is equal to the probability of seeing $Q$ in $DB$, and the denominator $Pr(p_{t_c^4})$ is equal to the probability of seeing $A\ overlaps\ B$. This probability is calculated by counting the number of times that $A\ overlaps\ B$ is followed by $A\ before\ C$ and $B\ before\ C$ (i.e., $Q$) in $DB$ and dividing it by the number of times we see $A\ overlaps\ B$ in $DB$.

Similar computations can be carried out at time points $t_c^1$ and $t_c^3$, but the situation is more complex since the time points are located after a starting point and before an ending point of an STI. Thus, since an STI that is not finished yet is involved, $p_{t_c}$ and $s_{t_c}$ cannot be described with Allen's temporal relations. Instead, we need to use a different representation based on STIs' *tieps* (Definition 2). Recalling that $A+$ denotes the starting *tiep* of an STI $A$ and $A-$ its ending *tiep*.

At $t_c^1$, the prefix $p_{t_c^1}$ is "$A$ that has started but not ended yet," and the suffix $s_{t_c^1}$ is "$B$ starts, $A$ ends, $B$ ends, $C$ starts, and then $C$ ends." Accordingly, $p_{t_c^1} = A+$ and $s_{t_c^1} = B+ < A- < B- < C+ < C-$. Thus, $Pr(Q \mid t_c^1)$ is equal to $Pr(Q)/Pr(A+)$, where $Pr(A+)$ denotes the probability of seeing STI $A$ that has started in $DB$. In practice, in the database $DB$, each STI has its starting and ending *tieps*, and thus, the probability $Pr(A+)$ equals the probability

of seeing STI $A$ in $DB$ ($Pr(A+)=Pr(A)$). Therefore, the probability $Pr(Q)/Pr(A)$ is calculated by counting the number of times we see TIRP $Q$ in $DB$ and dividing it by the number of times we see $A$ in $DB$.

Similarly, at $t_c^3$, the $p_{t_c^3}$ can be represented by the following inequality between the STIs' *tieps*: $A+ < B+ < A-$, and $s_{t_c^3}$ by $B- < C+ < C-$. However, since STI $B$ has started but not ended yet, its ending *tiep* $B-$ has to satisfy $t_c^3 < B-$ in database $DB$ (i.e., in the learning stage). Thus, the prefix's *tiep* ordering should be extended to $p_{t_c^3} = A+ < B+ < A- < B-$ to represent that STI $B$ is not ended. Otherwise, counting the number of times $p_{t_c^3}$ occurs in $DB$ might result in irrelevant instances, such as instances that $B-$ occurred before or co-occurred with $A-$, which are not part of the pattern's evolution process. The extended $p_{t_c^3}$ is equivalent to Allen's temporal relation $A$ *overlaps* $B$ (see Fig. 1, right-hand column). Therefore, the probability $Pr(Q \mid t_c^3) = Pr(Q)/Pr(A\ o\ B)$ is calculated, as computed at $t_c^4$.

In the previously discussed time points, we were able to eventually express the TIRP's prefixes using Allen's temporal relations. However, this may not be possible at other time points where multiple STIs have started but not yet ended. For example, at time point $t_c^2$, $p_{t_c^2}$ is equivalent to $A+ < B+$, and $s_{t_c^2}$ to $A- < B- < C+ < C-$. Since $p_{t_c^2}$ includes STIs $A$ and $B$ that have already started but not yet ended (i.e., $t_c^2 < A-$ and $t_c^2 < B-$), it results in uncertainty about which temporal relation between $A$ and $B$ will finally unfold. As shown in Fig. 1, based on $A$ and $B$'s starting *tieps* ($A+ < B+$), three different temporal relations are possible: *overlaps*, *contains*, or *finished-by*, which should be considered and used in Formula 3. Since this situation happens when the STIs are coinciding and not finished yet, we call it the *unfinished coinciding STIs* challenge and explain it in more detail in the following subsection.

Note that since the suffix of a TIRP at $t_c$ is not required for the TIRP's completion computations, this paper focuses on the prefixes of a TIRP.

### 3.3 The unfinished coinciding STIs challenge

We begin this subsection by formally introducing the notion of an unfinished STI.

**Definition 5** An *unfinished STI* $I^*$ at time $t_c$ is an STI whose starting *tiep* $I^*+$ satisfies $0 \leq I^*+ \leq t_c$ and whose ending *tiep* $I^*-$ satisfies $t_c < I^*-$.

The asterisk (*) will indicate that an STI is unfinished throughout the text. The start-time of an unfinished STI is known at time $t_c$, but its end-time is not. In fact, it is censored: we only know that it is later than $t_c$.

As we demonstrated in the previous subsection (Sect. 3.2), predicting the future temporal relations of coinciding STIs is trivial when one of the STIs' complete information is known, and no distribution or learning process is needed. However, the problem occurs when there are many possible temporal relations. A few scenarios according to which their Allen's temporal relation can evolve for a given pair of unfinished coinciding STIs $A^*$ and $B^*$. With no loss of generality, the starting *tieps* of the two unfinished STIs $A^*$ and $B^*$ satisfy either $A^*+ = B^*+$ or $A^*+ < B^*+$. To denote those situations, we introduce two additional "temporary" temporal relations between unfinished STIs, illustrated in Fig. 5. The *temporary equals* ($\stackrel{\check{}}{=}$) represents the temporal relation between a pair of unfinished coinciding STIs $A^*$ and $B^*$ having the same start-time $A^*+ = B^*+$ and is denoted by $A^* \stackrel{\check{}}{=} B^*$. When $A^*+ < B^*+$, the temporal relation will be called *temporary finished-by* (f̌i) and denoted with $A^*\,\check{\text{fi}}\,B^*$.

As shown in Fig. 5, a pair of unfinished coinciding STIs $A^*$ and $B^*$ may evolve into three possible temporal relations. The logic follows from the *tieps* representation of Allen's

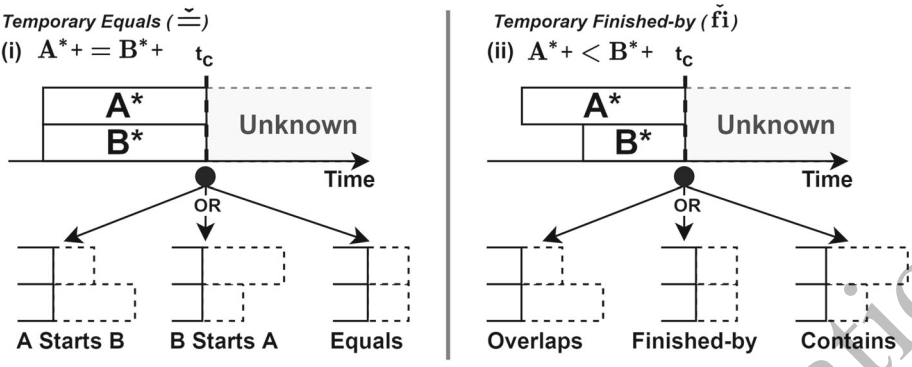



**Fig. 5** The possible temporary temporal relations between unfinished STIs, given that the start-times of a pair of STIs are known, but their end-times are not

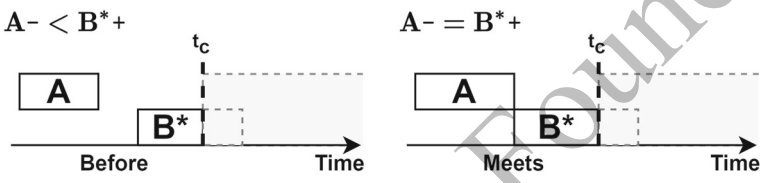



**Fig. 6** The possible temporal relations between a finished STI and an unfinished STI

temporal relations that is presented in Fig. 1 in the right-hand column. Figure 5.i shows that in the case of the *temporary equals* temporal relation, their temporal relation may eventually evolve into *A starts B*, or *B starts A*, or stay *A* equals *B*. The reason that *A starts B* and *B starts A* cannot be distinguished at $t_c$ is that the exact temporal relation is determined by their end-times, which are not yet known. Similarly, the *temporary finished-by* temporal relation shown in Fig. 5.ii, may eventually evolve into three possible Allen's temporal relations: *A overlaps B*, *A contains B*, or stay *A finished-by B*.

For completeness, even though they are not considered part of the unfinished coinciding STIs challenge, we mention two more scenarios, when the STIs' start-times and the earlier STI's end-time are known. As shown in Fig. 6, when *A* ends before or simultaneously as $B^*$ starts ($A^- \leq B^*+$), the temporal relations *before* or *meets* are produced without ambiguity. The problem described earlier does not occur since it is not a pair of unfinished STIs.

## 4 Methods

This study investigates the task of continuous prediction of a TIRP's completion, which was introduced in Sect. 3. This section presents a way to overcome the unfinished coinciding STIs challenge (Sect. 3.3) by defining a structure called TIRP-prefix (Sect. 4.1). The TIRP-prefixes will represent the evolution process of a TIRP and will be used to calculate the TIRP's completion probabilities by considering all possible TIRPs that each TIRP-prefix can evolve to.

Figure 7 illustrates the overall TIRP's completion model learning and predicting process. The TIRP's completion learning process (Fig. 7.I) gets a TIRP of interest and peels the TIRP into its TIRP-prefixes (Sect. 4.1). Then, the TIRP-prefixes' instances are detected from the

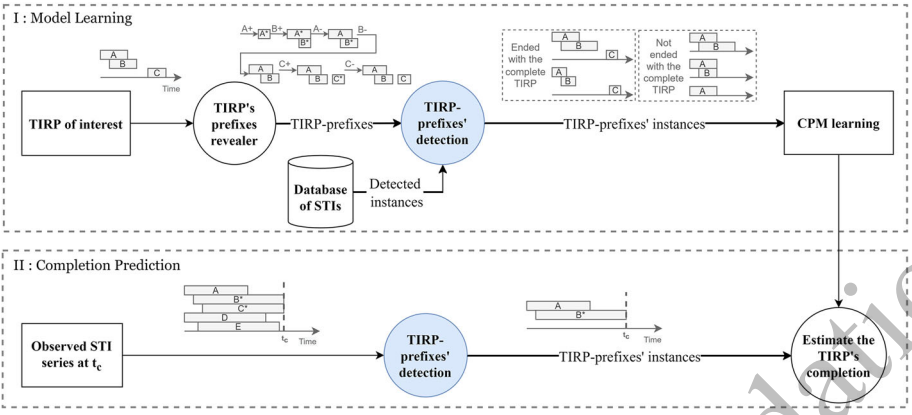



**Fig. 7** The overall TIRP's completion (I) model learning and (II) prediction

STIs database (Sect. 4.2-4.3) and used to learn a continuous prediction model (CPM, Sect. 4.4). The TIRP's completion prediction process (Fig. 7.II) gets an STI series and detects the TIRP-prefixes' instances (Sect. 4.3). At any time point, each detected TIRP-prefix's instance can be used to estimate the TIRP's completion probability using the learned CPM. Based on these probabilities, the early warning strategies (Sect. 4.5) can be used to raise an alert once there is a high likelihood that the pattern of interest will complete.

### 4.1 TIRP unfolding over time

Understanding the process of a TIRP's unfolding over time is necessary for continuous prediction of the pattern's completion since only part of the pattern is revealed at each time point. Since the temporal relation of unfinished coinciding STIs is undetermined until at least one of their ending *tieps* is observed, we follow the process of the TIRP unfolding over time by means of its STIs *tieps* (Definition 2). The TIRP-prefixes will be used to calculate the TIRP's completion probabilities by considering all possible TIRPs that each TIRP-prefix can evolve to.

Based on Definition 2, an STI $A$ can be represented by the starting *tiep* $A+ = (A_s, +, "A")$ and the ending *tiep* $A- = (A_e, -, "A")$. Consequentially, a TIRP also can be represented by starting and ending *tieps* since it is composed of a series of STIs. To express the temporal relations among the STIs in the series of *tieps*, the order among the starting and ending *tieps* should be considered. Thus, to maintain the conjunction of pairwise temporal relations among the STIs, the set of *tieps* needs to be transformed into a sorted tieps' series, based on Allen's *tieps* representation (Fig. 1, in the right-hand column). For example, the temporal relation *contains* between STIs $A$ and $B$ should satisfy the condition $A+ < B+ < B- < A-$, as illustrated in Fig. 1. The sorted *tieps* series corresponding to a TIRP $Q$ will be denoted with $Q_{tieps}$.

For the temporal relations *meets*, *finished-by*, *starts*, or *equals* between a pair of STIs, their *tieps* may co-occur and are merged into a single element in the *tieps*' series. For example, in Fig. 8, the temporal relation between STIs $B$ and $C$ is *meets*, which means STI $B$'s ending *tiep* $(B-)$ co-occur as STI $C$'s starting *tiep* $(C+)$. Since $B-$ and $C+$ co-occur, they are composed into a single element and wrapped in rounded brackets $(B-, C+)$.

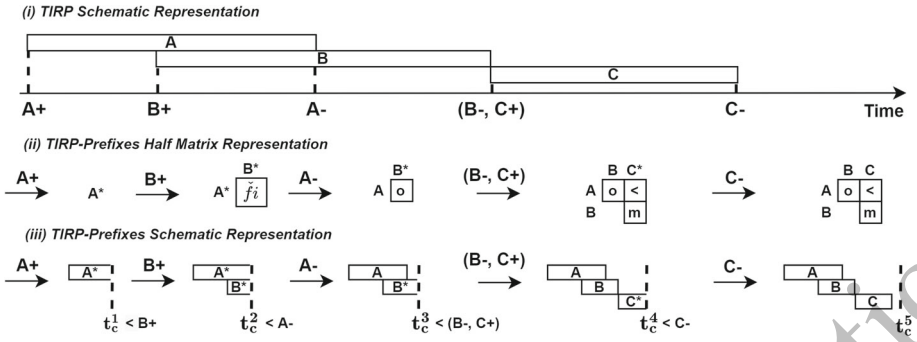



**Fig. 8** The TIRP: {*A overlaps B*, *A before C*, *B meets C*} and its TIRP-prefixes, presented in three ways: (i) TIRP schematic, (ii) TIRP-prefixes half matrix, and (iii) TIRP-prefixes schematic representation

We now formally define the notion of a TIRP-prefix, which constitutes a single step in the temporal evolution of a TIRP.

**Definition 6** Let $Q$ be a TIRP of length $k$. A *TIRP-prefix* $\breve{Q}$ of $Q$ is defined as a pair $\breve{Q} = (\breve{IS}, \breve{R})$, where $\breve{IS}$ is a lexicographical STI series (Definition 3) of $\breve{k} \leq k$ finished ($\breve{IS}_f$) and unfinished ($\breve{IS}_*$) STIs: $\breve{IS} = \breve{IS}_f \cup \breve{IS}_*$, and $\breve{R}$ is the set of all the temporal relations between each of the pairs of STIs in $\breve{IS}$: $\breve{R} = \{r(I^i, I^j) : 1 \leq i < j \leq \breve{k} \wedge \neg(I^i \in \breve{IS}_* \wedge I^j \in \breve{IS}_*)\} \cup \{\breve{r}(I^{*,i}, I^{*,j}) : 1 \leq i < j \leq \breve{k} \wedge I^{*,i} \in \breve{IS}_* \wedge I^{*,j} \in \breve{IS}_*\}$.

To capture the evolution of a given pattern and to be able to estimate the probability of its completion at any time point, all the stages in the process need to be considered. To do that, the pattern is divided into TIRP-prefixes (Definition 6) that are part of the TIRP's evolution process, which are created based on sub-sequences of the TIRP's *tieps*. In each TIRP-prefix, since the temporal relation between two unfinished STIs is uncertain, the temporary temporal relation $\breve{r}$ is used to express the disjunction of possible temporal relations based on the unfinished coinciding STIs challenge logic (Fig. 5).

In Fig. 8, the TIRP $Q$ = {A *overlaps B*, *A before C*, *B meets C*} is presented, with $Q_{tieps}$=<$A+$, $B+$, $A-$, ($B-$, $C+$), $C->$. Initially, only $A+$ is observed, which results in an unfinished STI $A^*$. Then, $B+$ appears, and another TIRP-prefix is revealed, including two unfinished STIs $A^*$ and $B^*$.

At time point $t_c^2$, it is already known that $A^*+ < B^*+ < t_c^2$, and since $A^*$ and $B^*$ are unfinished STIs, it can be concluded that $t_c^2 < A-$ and $t_c^2 < B-$. Thus, as presented in Fig. 5.ii, the temporal relation between $A^*$ and $B^*$ is temporary *finished-by* ($A^* \breve{\text{fi}} B^*$). At $t_c^3$, $A-$ is already known, while STI $B^*$ is unfinished ($t_c^3 < B-$), which means that the temporal relation is *overlaps*, since the temporal relation endpoint ordering $A+ < B+ < A- < B-$ is satisfied (Fig. 1). Similarly, at $t_c^4$, $B-$ and $C^*+$ co-occur, and the TIRP-prefix {*A overlaps B*, *A before* $C^*$, *B meets* $C^*$} is formed. The temporal relation *A before* $C^*$ is obtained since it is known that $A+ < A- < C+ < t_c^4 < C-$ and *B meets* $C^*$ ($B+ < B- = C+ < t_c^4 < C-$). Lastly, $C-$ reveals the entire pattern.

The TIRP-prefixes are used to learn a model intended to estimate the probability of the TIRP's completion at each time stamp (Sect. 4.4). Whenever a new *tiep* is encountered, the model uses the appropriate TIRP-prefix and provides the corresponding TIRP's completion probability, given the current time stamp. To create all TIRP-prefixes of a given TIRP during the learning stage, the TIRP-prefixes revealer algorithm (Algorithm 1) is introduced in the following subsection.

### 4.1.1 The TIRP-prefixes revealer algorithm

The TIRP-prefixes revealer algorithm (Algorithm 1) takes as input a TIRP of interest, which contains a series of STIs (*IS*) and conjunction of temporal relations between each pair of the STIs (*R*), and returns the TIRP-prefixes. Before describing the algorithm, we give an intuitive explanation of how the algorithm proceeds. The algorithm iterates through the pattern's *tieps* sequence in reverse order, considering in each iteration a single *tiep*, or more in case of co-occurring *tieps*. Based on the current iteration's *tieps*, the algorithm reveals the earlier TIRP-prefix by updating the current TIRP-prefix, considering, if necessary, the unfinished coinciding STIs challenge (Sect. 3.3). Although a forward-based algorithm could also be applied, in which each *tiep* is added, we think that starting with the entire TIRP and revealing the TIRP-prefixes in a backward order is more intuitive for explaining the process. For example, in Fig. 8, in the first iteration, the last *tiep* ($C-$) is used to create the previous TIRP-prefix. Initially, the current TIRP-prefix is defined as the entire TIRP, and since $C-$ is an ending *tiep*, STI *C* converted into an unfinished STI C*, and results in the previous TIRP-prefix: {A *overlaps B*, *A before C**, *B meets C**}. The algorithm stops once there are no more *tieps* to iterate and returns a list that contains all the TIRP-prefixes of the given TIRP.

**Algorithm 1** TIRP-Prefixes Revealer

**Input**: TIRP - composed of a series of STIs and a conjunction of temporal relations among the STIs.
**Output**: TIRPrefixes - sorted list of the TIRP-prefixes.

```
1: TIRPPrefixes ← [ ]
2: revTieps ← getReversedTieps(TIRP)                          ▷ reverse order of the TIRP's tieps.
3: currPrefix ← TIRP
4: for each currTieps in revTieps do                          ▷ currTieps is an element with at least one tiep
5:     for each tiep in currTieps do
6:         currentSTIsIndex ← getIndexOfCurrentSTI (currPrefix, tiep)
7:         if tiep.type == '+' then
8:             removeSTI(currPrefix, currentSTIsIndex)
9:         else
10:            addAsteriskToSymbol(currPrefix, currentSTIsIndex)          ▷ e.g., STI I to I*
11:            otherUnfSTIs ← getOtherUnfSTIs(currPrefix, currentSTIsIndex)
12:            for each UnfSTI in otherUnfSTIs do
13:                currRel = getRel(currentSTIsIndex, UnfSTI)
14:                if currRel == ('overlaps' or 'finished-by' or 'contains') then
15:                    changeRel(currentSTIsIndex, UnfSTI, 'temporary finished-by')
16:                else if currRel == ('starts' or 'equals') then
17:                    changeRel(currentSTIsIndex, UnfSTI, 'temporary equals')
18:    TIRPPrefixes.appendLeft(currPrefix)                        ▷ add by value
19: return TIRPPrefixes
```

The function *getReversedTieps* (Algorithm 1, line 2) gets a TIRP and sorts its *tieps'* sequence in reversed order of their time stamps. The reverse order sequence, returned by the function *getReversedTieps*, is assigned to the variable *revTieps*. Each element in *revTieps* contains a single *tiep*, or more in case of co-occurring *tieps*. For example, if $Q_{tieps}$ is equal to $<A+, B+, A-, (B-, C+), C->$, the reversed sequence *revTieps* is equal to $<C-, (B-, C+), A-, B+, A+>$. The variable *currPrefix* represents the current TIRP-prefix through each iteration and is used to reveal the previous TIRP-prefix. It is initialized in line 3 with the TIRP of interest.

The algorithm iterates through each element of *revTieps*, with the current element stored in *currTiep*, and through each *tiep* in *currTiep* (lines 4–18). In each iteration, the algorithm assigns the original *tiep*'s STI index in *currPrefix* to *currentSTIsIndex* using the function *getIndexOfCurrentSTI*.

Next, if the current *tiep* is a starting *tiep* (tiep.type is '+'), the function *removeSTI* removes the current *tiep*'s STI from the current prefix *currPrefix* (line 8). Otherwise, if the current *tiep* is an ending *tiep* (tiep.type is '−'), the algorithm adds an asterisk to the *tiep*'s STI's symbol, indicating an unfinished STI (line 10). Moreover, the algorithm establishes the temporal relations of the current unfinished STI with all other unfinished STIs in the current prefix, according to the unfinished coinciding STIs challenge (Sect. 3.3). Lastly, at the end of each iteration (line 18), the algorithm adds the *currPrefix* into *TIRPPrefixes*, which is the returned list of TIRP-prefixes.

Given a pattern having $k$ STIs, the runtime complexity of the TIRP-prefixes revealer algorithm is $O(k+k(k-1)) = O(k^2)$ due to the iterations over $k$ starting *tieps* and $k$ ending *tieps*, in which for each ending *tiep* iterates over up to $k-1$ other unfinished STIs.

### 4.2 Generation of The TIRP-prefix's evolving TIRPs

A TIRP-prefix containing more than one unfinished STI results in uncertain temporal relations (Sect. 3.3). As a result, all possible TIRPs that can evolve from the TIRP-prefix need to be considered for the computations. The TIRP-prefix's temporary disjunctions of temporal relations are replaced based on the rules presented in Fig. 5 to generate all possible combinations.

These combinations represent a set of TIRPs that can evolve from a current TIRP-prefix. For example, given the TIRP-prefix {$A^*$ temporary *finished-by* $B^*$}, the three following TIRPs can be evolved: {$A$ *overlaps* $B$} or {$A$ *finished-by* $B$} or {$A$ *contains* $B$}.

A naive generation of all possible patterns that can evolve from a given TIRP-prefix involves the enumeration of all possible temporal relations between each pair of unfinished STIs according to the rules shown in Fig. 5. For two or more unfinished coinciding STIs ($k \geq 2$), there are $(k^2-k)/2$ possible temporal relations into which the unfinished STIs can eventually evolve. Thus, the number of possible temporal relations candidates is bounded by $3^{(k^2-k)/2}$ for $k$ unfinished coinciding STIs. The number three, which is the base of the exponent in the upper bound, represents the number of possible temporal relations between each pair of unfinished STIs, as explained earlier and shown in Fig. 5. For example, having four unfinished coinciding STIs ($k=4$), $A^*$, $B^*$, $C^*$, and $D^*$, where $A^+ < B^+ < C^+ < D^+$, results in $(4^2-4)/2 = 6$ pairs of the unfinished STIs. The temporal relations *overlaps*, *finished-by*, or *contains* can evolve between a pair of unfinished STIs (Fig. 5.ii). Thus, a naive generation of all the suitable temporal relations among them requires generating up to $3^{(4^2-4)/2} = 3^6 = 729$ patterns.

However, such a naive generation may produce impossible patterns with combinations of temporal relations that contradict each other. For example, based on Allen's transition table [26], once a combination of STIs includes $A$ *overlaps* $B$ and $B$ *overlaps* $C$, the temporal relation between $A$ and $C$ cannot be *finished-by* or *contains*, but only *overlaps*. Such patterns cannot exist in reality, and detecting them is useless and time-consuming. Using the transitivity property [26] will reduce the number of generated candidates by avoiding impossible patterns, which we describe in detail in Appendix A to ease the reading of this section and due to space constraints. In the example presented in the previous paragraph,

a generation using the transitivity property generates only 103 patterns instead of the 729 patterns that the naive generation produces.

### 4.3 TIRP-prefixes' detection

To learn a continuous TIRP's completion prediction model (Sect. 4.4), the TIRP-prefixes' instances have to be detected in the STIs database $DB$, as explained in Sect. 3. However, since $DB$ comprises finished STIs, TIRP-prefixes with unfinished STIs cannot be detected in the $DB$. A TIRP-prefix with a single unfinished STI has no ambiguity in the temporal relations, and its instances can be detected in the $DB$ by assuming the single unfinished STI is finished, which means the remaining ending *tiep* is followed by the TIRP-prefix's latest *tiep*. However, for a TIRP-prefix with multiple unfinished STIs, each TIRP that can evolve from the TIRP-prefix (Sect. 4.2) should be detected separately, which will be referred to in the detection algorithm also as TIRP-prefix.

The sequential TIRP-prefix detection algorithm (Algorithm 2) is used to detect TIRP-prefix instances in the STIs database. The sequential TIRP-prefix detection algorithm gets a TIRP-prefix to detect (*TIRPPrefix*) and lexicographically ordered STIs of an entity (e.g., patient), represented by *entSTIs*, and returns all TIRP-prefix's instances that were detected in this entity. The algorithm starts by detecting all instances of the TIRP-prefix's first STI in a given entity and expands these instances by detecting instances of further TIRP-prefix's STIs while verifying the instances' temporal relations are the same as in the TIRP-prefix's temporal relations half matrix. Only the detected instances that maintain the TIRP-prefix's conjunction of pairwise temporal relations among the STIs are returned.

**Algorithm 2** Sequential TIRP-Prefix Detection

**Input**: *TIRPPrefix* - a TIRP-prefix to detect; *entSTIs* - entity's lexicographically ordered STIs.
**Output**: *entTIRPPrefixsInst* - detected instances of the TIRP-prefix in the entity.

```
1: entTIRPPrefixsInst ← [ ]
2: for each currSTI in TIRPPrefix.STIs do
3:     i ← getSTIIdx(TIRPPrefix, currSTI)                          ▷ returns STI index
4:     currTIRPPrefixsInst ← [ ]
5:     entSTIInst ← getEntSTIInst(entSTIs, currSTI)                ▷ returns all detected instances
6:     if i == 0 then                                              ▷ TIRP-prefix's first STI
7:         currTIRPPrefixsInst ← entSTIInst
8:     else
9:         for each currEntSTI in entSTIInst do
10:            for each prevEntInst in currTIRPPrefixsInst do
11:                validRel ← True
12:                for each prevEntInstSTI in prevEntInst do
13:                    j ← getSTIIdx(TIRPPrefix, prevEntInstSTI)
14:                    expectedRel ← getExpectedRel(TIRPPrefix, i, j)
15:                    actualRel ← getActualRel(prevEntInstSTI, currEntSTI)
16:                    if actualRel ! = expectedRel then           ▷ different temporal relations
17:                        validRel ← False
18:                        break
19:                if validRel == True then                        ▷ temporal relations conjunction verification
20:                    newInst ← prevEntInst.append(currEntSTI)
21:                    currTIRPPrefixsInst.append(newInst)
22:    entTIRPPrefixsInst ← currTIRPPrefixsInst
23: return entTIRPPrefixsInst
```

In Algorithm 2, the detected TIRP-prefix's instances are represented by the variable *entTIRPPrefixsInst* and initiated to an empty list (line 1). The algorithm starts by iterating through the TIRP-prefix's STIs in their lexicographical order, the variable *currSTI* representing the current STI that needs to be detected (line 2). The function *getSTIIdx* is called to get the *currSTI*'s index in the TIRP-prefix, which is assigned to a variable $i$ (line 3). In line 4, the variable *currTIRPPrefixsInst* is initiated to an empty list and is used to collect all instances that include the TIRP-prefix's STIs with an index equal to or less than $i$. In each instance in *currTIRPPrefixsInst*, the temporal relations among the TIRP-prefix's STIs are the same as in the original TIRP-prefix. Then, in line 5, the function *getEntSTIInst* is used to get all instances of the current STI detected in *entSTIs*. They are stored in *entSTIInst*.

In case the current STI is the TIRP-prefix's first STI ($i = 0$), the variable *currTIRPPrefixsInst* is set to be *entSTIInst* (lines 6–7). Otherwise ($i > 0$), the temporal relations of each of *entSTIInst*'s instances (represented by *currEntSTI*), and each *currTIRPPrefixsInst*'s STIs instance, represented by *prevEntInst*, are compared. The temporal relation between *currEntSTI* and *prevEntInst* has to be verified to ensure it is the same as in the TIRP-prefix's temporal relations conjunction (lines 8–21).

The verification is performed by comparing each *currEntSTI* with each *prevEntInst*'s STI, which is represented by *prevEntInstSTI*. The result of this verification is stored in the variable *validRel*, which indicates whether the temporal relations among the detected TIRP-prefix's STIs are the same as in the TIRP-prefix's temporal relations conjunction. In each iteration of the verification, the algorithm executes the following steps: (a) the function *getExpectedRel* gets a *TIRPPrefix* and indices of two TIRP-prefix's STIs and returns the temporal relation between them. This function is called to get the expected temporal relation between *currEntSTI* and *prevEntInstSTI*, and the variable *expectedRel* stores its output (line 14); (b) the function *getActualRel* gets two detected STIs instances and returns the temporal relation between them, i.e., the actual temporal relation between *currEntSTI* and *prevEntInstSTI*, stored in *actualRel* (line 15); (c) then, if the *actualRel* was always equal to the *expectedRel*, the *prevEntInst* instance is expanded to also contain *currEntSTI* and added to *currTIRPPrefixsInst* (lines 16–21); (d) lastly, once the iterations over the *entTIRPPrefixsInst* are completed, *currTIRPPrefixsInst* is assigned to *entTIRPPrefixsInst* to store all expanded instances that include TIRP-prefix's STIs with indices less than or equal to $i$.

All TIRP-prefixes' instances must be detected from the training data to learn a CPM. In contrast to the learning stage, in which the uncertainty is problematic for the computations, in the prediction stage, the exact TIRP-prefix, even if included multiple unfinished STIs, should be detected to provide the appropriate TIRP's completion probability. A TIRP-prefix can be detected directly from the entity's data during the prediction stage since the entity's data contain unfinished STIs and temporary temporal relations.

Thus, the sequential TIRP-prefix detection (Alg. 2) can be used without any changes, during the prediction stage, at any time point, by passing the TIRP-prefix and the entity's observed data to the algorithm. A TIRP-prefix can be passed as a *TIRPPrefix*, and the entity's observed data as *entSTIs*, including the temporary relations in both variables. For example, the TIRP-prefix $\{A^*$ temporary *finished-by* $B^*\}$ can be detected in an entity only if there are unfinished STIs $A^*$ and $B^*$ and their temporary temporal relation is *finished-by* (i.e., $A^*+ < B^*+$).

Given a TIRP-prefix $\breve{Q}$ having $k$ STIs to be detected within an entity, having $n$ STIs, and up to $m$ instances that exist for each TIRP-prefix's STI in the entity, the sequential TIRP-prefix detection algorithm is bounded in the worst case by the following expression: $O(k{*}n + k{*}m^k) = O\big(k(n + m^k)\big)$. The algorithm iterates through the $k$ TIRP-prefix's STIs and goes over the $n$ entity's STIs to detect the current TIRP-prefix's STI (line 5). Additionally,

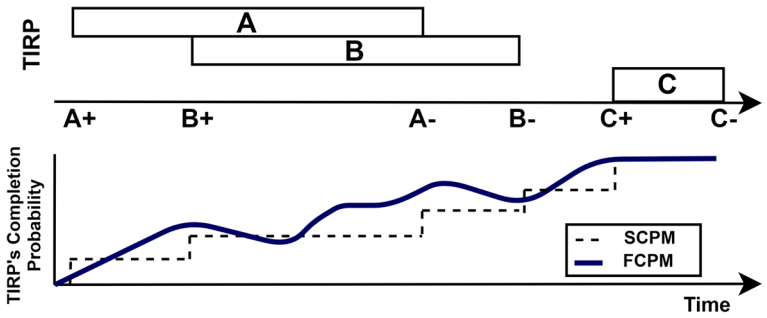



**Fig. 9** An example of a TIRP with three STIs (top) and the predicted TIRP's completion probabilities (bottom) using the two continuous prediction models: SCPM and FCPM

in each iteration, the algorithm verifies the instances of TIRP-prefix's STI $i$ with the STIs in *currTIRPPrefixsInst* (lines 6–21). Overall, the number of runs for the temporal relations verification is $m+1*m^2+2*m^3+\cdots+(k-1)*m^k$. The constant change in the expression's elements represents the number of comparisons required for the TIRP-prefix's temporal relations in *currTIRPPrefixsInst*. The multiplying in $m$ represents a case that in each iteration on the TIRP-prefix's STIs, the instances in *currTIRPPrefixsInst* were verified to ensure they have the same temporal relations as in $Q$'s conjunction of temporal relations. Lastly, this expression can be upper bounded by $k*m^k = k(m+m^2+m^3+\cdots+m^k)$.

### 4.4 Continuous prediction models

We introduce two continuous prediction models (CPMs) to continuously estimate the probability of the TIRP's completion based on the TIRP-prefixes. The first, more simplistic, model is called the *Segmented Continuous Prediction Model (SCPM)*. The predicted TIRP's completion probability changes only at time points where *tieps* appear, as illustrated by a dashed line at the bottom of Fig. 9. The second model, called the *Fully Continuous Prediction Model (FCPM)*, is more complex and incorporates the durations between the TIRP-prefix's consecutive *tieps* and thus allows the estimated completion probability to change continuously, as shown by the solid line at the bottom of Fig. 9.

#### 4.4.1 The segmented continuous prediction model

A basic rule used to calculate the completion probability was already laid out in Sect. 3.1, Formula 3. Recall that the probability of TIRP $Q$ completion is $Pr(Q)/Pr(p_{t_c})$, where $p_{t_c}$ is $Q$'s prefix at time $t_c$. Thanks to the TIRP-prefixes definition and their instances' detection, such probabilities can be computed, as explained in Sect. 4.3.

The *Segmented Continuous Prediction Model (SCPM)* is based on simply applying Formula 3, and since it was discussed in detail in Sects. 3.1 and 3.2, it will not be discussed further.

#### 4.4.2 The fully continuous prediction model

The *Fully Continuous Prediction Model (FCPM)* considers the distributions of the durations between the TIRP-prefixes' consecutive *tieps* (Fig. 10), in addition to the TIRP-prefixes'

number of occurrences in the $DB$. The FCPM uses these distributions to estimate the TIRP's completion given the durations between the *tieps* in the TIRP-prefix $p_{t_c}$ of $Q$ observed at any time point $t_c$.

In this section, the durations between consecutive *tieps* of a TIRP-prefix are considered a *duration element* of $p_{t_c}$, and its estimated distribution will be used to calculate the TIRP's completion probability. Let $d_i$ be the duration of the TIRP-prefix's $i$-th element, and $g$ the total number of duration elements in $p_{t_c}$. The SCPM's Formulas 1, 2, and 3 are modified to include durations as follows[2]:

$$Pr(Q \mid t_c, d_1, \ldots, d_g) = Pr(s_{t_c} \mid p_{t_c}, d_1, \ldots, d_g) \tag{5}$$

$$= \frac{Pr(p_{t_c}, d_1, \ldots, d_g \mid s_{t_c}) Pr(s_{t_c})}{Pr(p_{t_c}, d_1, \ldots, d_g)} \tag{6}$$

$$= \frac{Pr(d_1, \ldots, d_g \mid p_{t_c}, s_{t_c}) Pr(p_{t_c} \mid s_{t_c}) Pr(s_{t_c})}{Pr(d_1, \ldots, d_g \mid p_{t_c}) Pr(p_{t_c})} \tag{7}$$

$$= \frac{Pr(d_1, \ldots, d_g \mid Q) Pr(Q)}{Pr(d_1, \ldots, d_g \mid p_{t_c}) Pr(p_{t_c})}. \tag{8}$$

Formula 5 gives the TIRP $Q$'s completion probability $Pr(Q \mid t_c)$ estimated at time point $t_c$ taking into account the durations distributions between the TIRP-prefix's consecutive *tieps*. The equality in Formula 5 results from observing $Q$ is equivalent to observing the prefix $p_{t_c}$ at $t_c$, later followed by $s_{t_c}$.

Formula 6 follows the Bayes' rule, and in Formula 7, the numerator results from the conditional probability rule applied conditional on $s_{t_c}$. The denominator in Formula 7, again follows from the conditional probability rule. Lastly, in Formula 8, the numerator is derived based on the conditional probability rule $Pr(p_{t_c} \mid s_{t_c}) Pr(s_{t_c}) = Pr(p_{t_c}, s_{t_c})$, and observing $p_{t_c}$ followed by $s_{t_c}$ is equivalent to observing $Q$.

In full generality, the duration elements can be dependent on each other and follow a complex multidimensional continuous distribution. To be able to use the duration elements in a practical model, we make the following simplifying assumptions:

**Assumption 1** The duration elements $d_1, \ldots, d_g$ are independent conditionally on the event $(p_{t_c}, s_{t_c})$, which is equivalent to $Q$.

**Assumption 2** The duration elements $d_1, \ldots, d_g$ are independent conditional on the event $(p_{t_c}, \neg s_{t_c})$.

Intuitively, we assume the duration elements to be independent at every time point $t_c$ in both TIRP-prefix's instances that are ended or not ended with the complete TIRP $Q$. Based on Assumptions 1 and 2, Formula 8 can be rewritten as:

$$Pr(Q \mid t_c) = \frac{\psi(t_c)}{\psi(t_c) + \bar{\psi}(t_c)}, \tag{9}$$

[2] Note that in the formulas we use probabilities of continuous variables. The issue will be discussed in detail below, for now let us assume it to be a shorthand notation for probabilities of durations belonging to a narrow interval around the current value.

where

$$\psi(t_c) = Pr(p_{t_c}, s_{t_c}) \prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, s_{t_c}), \tag{10}$$

$$\bar{\psi}(t_c) = Pr(p_{t_c}, \neg s_{t_c}) \prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, \neg s_{t_c}). \tag{11}$$

The numerator in Formula 9, which is defined in Formula 10, results from applying Assumption 1 to the numerator of Formula 8: $Pr(Q)Pr(d_1, \ldots, d_g \mid Q)$, which is equivalent to $Pr(p_{t_c}, s_{t_c})Pr(d_1, \ldots, d_g \mid p_{t_c}, s_{t_c})$. Based on Assumption 1, the multiplication law of probability is used on $Pr(d_1, \ldots, d_g \mid p_{t_c}, s_{t_c})$ resulting in $\prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, s_{t_c})$. The denominator in Formula 9 is obtained based on Assumption 2 by first applying the conditional law of total probability:

$$Pr(d_1, \ldots, d_g \mid p_{t_c}) = \tag{12}$$

$$= Pr(d_1, \ldots, d_g \mid p_{t_c}, s_{t_c})Pr(s_{t_c} \mid p_{t_c}) + Pr(d_1, \ldots, d_g \mid p_{t_c}, \neg s_{t_c})Pr(\neg s_{t_c} \mid p_{t_c})$$

$$= \prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, s_{t_c}) \frac{Pr(p_{t_c}, s_{t_c})}{Pr(p_{t_c})} + \prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, \neg s_{t_c}) \frac{Pr(p_{t_c}, \neg s_{t_c})}{Pr(p_{t_c})} \tag{13}$$

$$= \frac{1}{Pr(p_{t_c})} \Big[ Pr(Q) \prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, s_{t_c}) + Pr(p_{t_c}, \neg s_{t_c}) \prod_{i=1}^{g} Pr(d_i \mid p_{t_c}, \neg s_{t_c}) \Big] \tag{14}$$

$$= \frac{\psi(t_c) + \bar{\psi}(t_c)}{Pr(p_{t_c})}. \tag{15}$$

Formula 12 follows from the conditional law of total probability, Formula 13 from Assumptions 1 and 2, and the definition of conditional probability. Formula 14 is obtained by refactoring, and Formula 15 follows from Formulas 10 and 11. Finally, the denominator in Formula 9 ($\psi(t_c) + \bar{\psi}(t_c)$) results from multiplying the previous result by $Pr(p_{t_c})$ in the denominator of Formula 8. The decomposition reduces the problem to estimating $2g$ single variable distributions for each possible TIRP-prefix of $Q$.

For the computations, we use Formula 9 ($\psi(t_c) / \left[\psi(t_c) + \bar{\psi}(t_c)\right]$), in which $\psi(t_c)$ is defined in Formula 10, and $\bar{\psi}(t_c)$ is defined in Formula 11. To estimate the duration probabilities involved in the TIRP-prefix, we fit a duration distribution for each duration element between the TIRP-prefix's consecutive *tieps*. This will be illustrated below.

For example, in Fig. 10, the TIRP-prefix $p_{t_c} = \{A \textit{ overlaps } B^*\}$ contains three *tieps*, which means $g$ is equal to three. The TIRP's completion probability estimation is based on duration histograms and fitted distributions between the consecutive *tieps* of the TIRP-prefix $\{A \textit{ overlaps } B^*\}$. As shown in Fig. 10.i, to calculate Formula 10, the instances of $\{A \textit{ overlaps } B^*\}$ that end with the complete TIRP in $DB$ should be considered, and for Formula 11, the instances of $\{A \textit{ overlaps } B^*\}$ which do not end with the complete TIRP in $DB$ should be used, as shown in Fig. 10.ii.

Some of the TIRP-prefix's instances that do not end with the complete TIRP might have different consecutive *tieps* than the consecutive *tieps* of the complete TIRP. For example, in Fig. 10, the TIRP's earliest consecutive *tieps* are $A+$ and $B+$. However, the earliest TIRP-prefix is $A^*$, whose instances need not necessarily satisfy the temporal relation *overlaps* with STI $B$. For these instances, the TIRP's consecutive *tieps* $A+$ and $B+$ do not exist; thus, the duration between the consecutive *tieps* $A+$ and $A-$ needs to be used to estimate $A$'s duration.

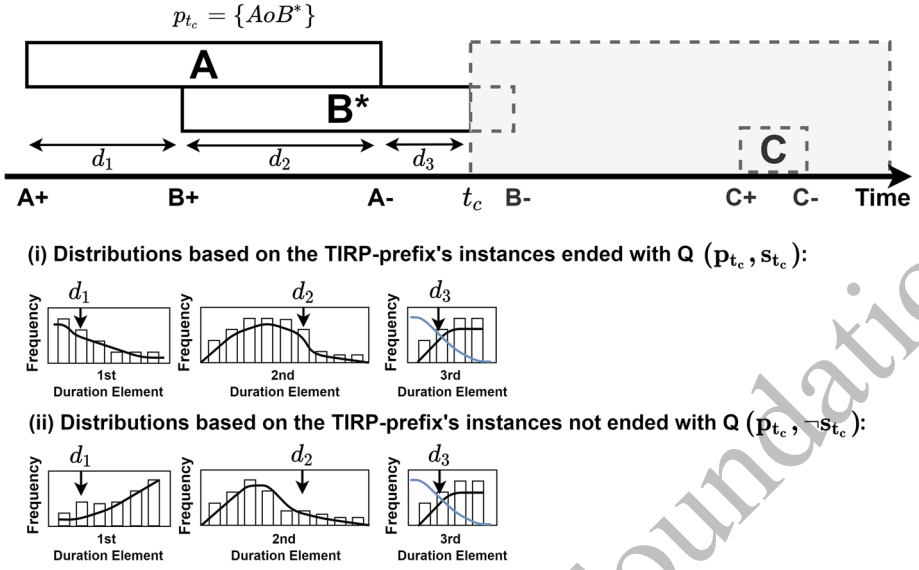



**Fig. 10** A continuous completion prediction of TIRP $Q$: {*A overlaps B*, *A before C*, *B before C*}, considering the durations distributions between the TIRP-prefix's consecutive *tieps* of TIRP-prefix's instances that are (i) ended or (ii) not ended with $Q$ in $DB$

*Distribution Estimation Procedure* We now describe the procedure used for actual distribution estimation. We assume that each element's duration follows one of the distributions commonly used in survival analysis: exponential, Weibull, gamma, or lognormal. Additionally, we include the Pareto distribution to allow heavy tails and the exponentiated Weibull distribution to allow the steep sides and a flat bottom density (i.e., the so-called "bathtub distribution"). The parameters of each distribution are fitted to the data using the maximum likelihood method. The final distribution is determined by constructing the histogram of duration values and choosing the distribution with the least residual sum of squares between its density function and the histogram. We used the distribution estimation procedures implemented in Python's SciPy package [47] (version 1.4.1), and the number of bins in the histogram was determined using the Freedman-Diaconis rule [48].

*Probability Mass* Each $d_i$ is a real number representing the duration, and thus, the probability of seeing this *exact* duration is zero. Thus, the probability cannot be provided for a specific value, only for a range. For notational simplicity, the event $d_i \in [d_i - \epsilon, d_i + \epsilon]$ for some constant $\epsilon > 0$ will be denoted simply with $d_i$ within probability statements. $\epsilon$ thus represents the width of the interval around the current value of $d_i$ in which we compute the probability mass. In our implementation, we found the value of $\epsilon = 1$ to be optimal, as smaller values would introduce too much noise into the estimated probabilities.

*Censoring* For the duration of the last and unfinished element of $p_{t_c}$ (i.e., $d_g$), the probability of it belonging to a narrow interval cannot be used in Formulas 10 and 11 since the exact duration is not yet known. The value is *censored* since we only know that the given STI will end later than $t_c$. Since the problem is analogous to censoring in survival analysis, we adopt a method used in this branch of statistics to handle censored data [49, Chapter 3]. Instead of using the probability density function for $d_g$, we will use the probability that the duration is longer than the currently observed time ($Pr(d_g \geq t_c)$), where the probability is computed according to the fitted distribution. As shown in Formula 16, the probability $Pr(d_g \geq t_c)$

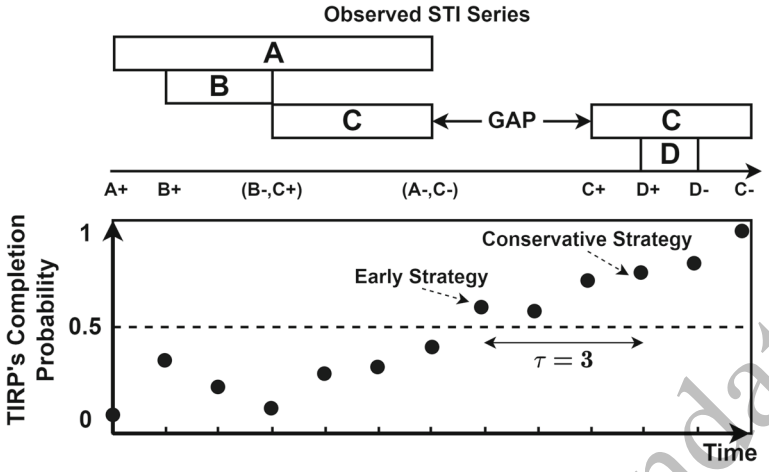



**Fig. 11** The horizontal dashed line indicates the prediction decision threshold (0.5 in this figure). An alert could be raised immediately after the probability exceeds the threshold (early strategy) or when the threshold was consistently exceeded for some pre-defined time $\tau$ (conservative strategy)

equals to the survival function $1 - CDF(t_c)$, which is one minus the cumulative distribution function (CDF) of $d_g$. For example, in Fig. 10, the blue-colored functions represent the $1 - CDF(t_c)$ for the third duration element.

$$Pr(d_g \geq t_c) = 1 - CDF(t_c). \tag{16}$$

When Formula 16 is used to calculate the probability of the duration of the last and unfinished element $d_g$, Formulas 10 and 11 are rewritten, respectively, as follows:

$$\psi(t_c) = Pr(p_{t_c}, s_{t_c}) Pr(d_g \geq t_c \mid p_{t_c}, s_{t_c}) \prod_{i=1}^{p-1} Pr(d_i \mid p_{t_c}, s_{t_c}), \tag{17}$$

$$\bar{\psi}(t_c) = Pr(p_{t_c}, \neg s_{t_c}) Pr(d_g \geq t_c \mid p_{t_c}, \neg s_{t_c}) \prod_{i=1}^{p-1} Pr(d_i \mid p_{t_c}, \neg s_{t_c}). \tag{18}$$

### 4.5 Early warning strategies

Early warning strategies can be used to raise an alert once there is a high likelihood of the completion of the TIRP, based on the CPMs' estimated probabilities (Fig. 9). For example, predicting a TIRP's completion using early warning strategies can be used for early prediction of a sequence of events ending with a target event by continuously predicting the completion of a TIRP ending with that event. In medicine, the clinical team needs to be warned of potential complications of a monitored patient in an ICU to enable early intervention, and ideally, prevention.

We propose two strategies to predict the TIRP's completion, as illustrated in Fig. 11. The *early strategy* triggers a warning when the completion probability crosses a pre-defined threshold.

Alternatively, the *conservative strategy* only issues an alert only when the probability crosses a pre-defined threshold for at least a pre-defined amount of time, called the *decision*

*time delay* ($\tau$). For example, in Fig. 11, $\tau$ is defined as three time stamps. The early strategy is a special case of the conservative strategy with a $\tau = 0$.

A lower $\tau$ might give earlier decisions and increase the absolute number of the true positive cases (TP) cases since it predicts the pattern will be unfolded once the probability crosses the threshold, even for a short time. However, a lower $\tau$ might also increase the false-positive (FP) cases since it can capture cases that cross the threshold by chance. In contrast, a higher $\tau$ leads to more conservative predictions, which may result in delayed decisions and increase the true-negative (TN) cases. However, for higher $\tau$, the false-negative (FN) cases might also increase since the patterns' completion might appear before the probability crosses the threshold for the required $\tau$ time units.

## 5 Evaluation

Our goal was to evaluate the effectiveness of using the continuous prediction models (CPMs) in predicting a TIRP's completion, which will be used in future work for predicting an event of interest that a pattern ends with. The evaluation was performed on four real-life medical and non-medical datasets, and three research questions were defined. Overall, the TIRP's completion evaluation using the CPMs was performed on over 2,000 different TIRPs.

The main *research questions* for this study were:

RQ1. Which CPM performs better, in terms of prediction performance and earliness, in predicting a TIRP's completion?
RQ2. Which value of $\tau$ performs better, in terms of prediction performance and earliness, in predicting a TIRP's completion?
RQ3. Which temporal abstraction method and number of symbols lead to better prediction performance of the CPMs for a given target event using TIRP's completion?

### 5.1 Datasets

We evaluated the proposed models using real-life datasets: the cardiac surgical patients (CSP) dataset (Sect. 5.1.1), acute hypertensive episodes (AHE) dataset (Sect. 5.1.2), diabetes (DBT) dataset (Sect. 5.1.3), and elderly first injury fall (EFIF) dataset (Sect. 5.1.4).

#### 5.1.1 The cardiac surgical patients (CSP) dataset

The cardiac surgical patients (CSP) dataset [9] contains data measured every minute over the first 12 h of the ICU hospitalization of patients who underwent cardiac surgery at the Academic Medical Center in Amsterdam, the Netherlands, from 2002 to 2004. The data include: mean arterial blood pressure (MAP), central venous pressure (CVP), heart rate (HR), body temperature (TMP), and two ventilator variables, namely fraction inspired oxygen (FiO2) and level of positive end-expiratory pressure (PEEP), and low-frequency time-stamped data, including base excess (BE), cardiac index (CI), creatinine kinase MB (CKMB), and glucose. For knowledge-based discretization, we created states for each temporal variable using cutoffs that knowledge experts defined in Verduijn [50].

The target event was defined as the first occurrence of CI with values lower than 2.5 ls per minute per square meter ($L/min/m^2$). The patient's files are filtered to at least 120 min of data before the target event, leaving the data to a total of 329 patients, of which 115 patients

had the target event in their records. PAA was used by dividing the time series into segments of ten minutes.

### 5.1.2 The acute hypertensive episodes (AHE) dataset

The acute hypertensive episodes (AHE) data [12] were extracted from the MIMIC-III [29], a public-access database that contains discrete patient electronic medical records (EMR) data derived from multiple ICUs from a single center in Boston, Massachusetts, USA. We used vital signs data, which are time series-based sampled each minute and comprised of systolic arterial blood pressure (SABP), heart rate (HR), arterial oxygen saturation (SpO2), and respiratory rate (RESP).

The target event was defined as the AHE target onset, which was defined as the beginning of overlapping epoch intervals of 30 min, including more than half of the data points with SABP greater than 130 mmHg. These episodes represent elevations in blood pressure and may result in clinical damage or indicate a change in the patient's clinical situation, such as elevation of intracranial pressure or kidney failure [12].

The actual 1:1 ratio between the patients who have and do not have the target event in their records was maintained, and the patients' similarity score was calculated by their demographic parameters. The patient's files are filtered to at least 12 h of data for each patient, leaving the data to a total of 2688 patients, of which 1344 patients had the target event in their records. PAA was used by dividing the time series into segments of one hour.

### 5.1.3 The diabetes (DBT) dataset

The diabetes dataset [9] provided by Clalit Health Services, an Israeli health maintenance organization, contains monthly data from 2002 to 2007 on type II diabetes patients. The dataset contains hemoglobin-A1c (HbA1C) values, blood glucose levels, low-density lipoprotein (LDL) cholesterol values, albumin values, and creatinine levels, and medications that the patients purchased: oral hypoglycemic agents (diabetic medications), cholesterol-reducing statins, and beta-blockers. For knowledge-based discretization, we created states for each temporal variable using cutoffs that knowledge experts defined, in which we used the definitions described in Moskovitch and Shahar [9].

The target event was defined as the first occurrence of high HbA1C with values greater than 9%. The patient's files are filtered to at least 24 months of data before the target event, leaving the data to a total of 1,710 patients, of which 239 patients had the target event in their records.

### 5.1.4 The elderly first injury fall (EFIF) dataset

The elderly first injury fall (EFIF) data [45] were collected between 2017 and 2019 from over 1769 care homes across the UK. The carers documented the data through a mobile app and contained their assessments of the residents' actions. The data were aggregated to a granularity of days and contained the following temporal variables: number of drinking actions, number of smoking actions, number of exercise actions, averaged estimated appetite levels, averaged estimated amount of fluid drunk, averaged estimated mobility assistance, averaged estimated happiness levels, average blood glucose levels, and number of routine medical actions (e.g., skin integrity assessment or vital signs measurements).

The target event was defined as the first occurrence of the residents' first fall with a severe or moderate injury. The resident's files are filtered to residents above 75 years old that are not bed bound and at least 6 weeks data before the target event, leaving the data to 852 residents, of which 150 residents had the target event in their records. The residents were matched based on their similarity score, calculated by their demographic parameters. PAA was used by dividing the time series into segments of two weeks.

### 5.2 Experiments

To answer the research questions, an experiment was designed and performed. Additionally, a preliminary analysis of the experiment was performed (Appendix E), in which the goal was to evaluate the CPMs at the instances' different portions over time, without using the early warning strategies.

#### 5.2.1 Experimental setup

The models were evaluated on the ability to predict the completion of a TIRP that ended with a target event. The entities' (i.e., patients) demographic data were ignored, and only the time-based data were used for the prediction.

Data-driven temporal abstraction methods (Sect. 2.1) were used, including SAX, EWD, and EFD, with two, three, and four symbols per variable. For gradient abstraction (GRAD), three symbols were used to represent the first derivative's sign: increasing, stable or decreasing symbols. The first derivative was calculated based on an overlapping time window of 60 min for the CSP dataset, 4 h for the AHE dataset, 5 months for the DBT dataset, and 2 months for the EFIF dataset. As specified in Appendix B, the knowledge-based (KB) abstraction was performed using cutoffs that a domain expert defined for the CSP, AHE, and DBT datasets. The STIs were concatenated into a unified STI when the gap among two STIs having the same symbol was shorter than 10 min for the CSP dataset, 1 h for the AHE dataset, 1 month for the DBT dataset, and 2 weeks for the EFIF dataset, and there were no other STIs between them from the same variable.

Each pattern that ended with the target event was used separately to learn a model for its completion. Then, the training set detected TIRP-prefixes instances were used to learn the CPM model and were evaluated on continuously predicting the completion of the TIRP-prefixes instances that were detected in the testing set. The patterns were discovered from the STI data using the KarmaLego algorithm [27] with Allen's seven temporal relations. The patterns were discovered in the data with a minimum vertical support threshold (Sect. 2.3) of 30% out of the entities (i.e., patients) that contained the target event. Out of all the discovered patterns, those that ended with the target event were used. The target events' STIs were considered as instantaneous events, in which the beginning of the target event was considered as the completion of the TIRP. Once the patterns were discovered, all entities, with or without the target event, were used for learning and evaluating the model.

A TIRP and its TIRP-prefixes can be detected more than once in an entity's records, in which each detected instance of the TIRP or its TIRP-prefixes were considered and evaluated separately. Overall, all instances that started with the TIRP's earliest *tiep*, or more in the case of co-occurring *tieps*, were used in the experiments. Since each TIRP's completion was based on a different number of detected TIRP-prefixes' instances that ended or not ended with the TIRP, the imbalanced ratio differed between the patterns (see Appendix C). We ran the experiments with ten-fold cross-validation, using target stratification. The instances of

a TIRP and its TIRP-prefixes of the same entity (e.g., patient) appeared exclusively in the same fold. As we explain in the description of the experiment (Sect. 5.2.2), the instances were evaluated at each time stamp as long as the model did not make a decision or until the end of the entity's data. Lastly, once the final decisions for the TIRP's completion were collected, we evaluated the models compared to the actual labels.

**Baseline models** In addition to the SCPM and FCPM, other models consisting of binary classifiers were evaluated, in which the durations between the consecutive TIRP-prefix's *tieps* were used as features. For that, random forest (RF) [51], artificial neural network (ANN) [52], and recurrent neural network (RNN) [53] classifiers were used.

Records for the classifier were created to represent the evolution of a TIRP over time. Each input record for the classifier represented a TIRP-prefix instance at a specific time stamp, in which multiple records were used to consider all the time stamps for each evolving TIRP-prefix instance. The TIRP-prefix instance's time durations between the consecutive *tieps* at a specific time stamp were used as features. Each instance's record target was whether the instance was finally unfolded to a TIRP's completion or not. For example, in Fig. 10, the presented instance of TIRP-prefix $\{A\ overlaps\ B^*\}$ is represented by the time duration elements $d_1$, $d_2$, and $d_3$. The TIRP's duration elements that were not observed until time point $t_c$ are represented by zero. In Fig. 10, the presented TIRP-prefix's instance at time point $t_c$ is an example of an instance that ended with the entire TIRP. The order between the time durations elements between the consecutive TIRP-prefix's *tieps* was considered for the recurrent neural network. More details on the settings of the baseline models are described in Appendix D.

Note that our evaluation aims to assess the effectiveness of using the CPMs in predicting the completion of a TIRP's instance. As event prediction was not the focus of this evaluation (but future use of this capability), no state-of-the-art sequential deep learning models were used as baseline models to be learned from the raw data for predicting the event.

### 5.2.2 Continuous TIRP's completion prediction

The experiment's goal was to evaluate the ability to make decisions over time regarding whether a TIRP will be unfolded to its completion using the CPMs (Sect. 4.4) and the early warning strategies (Sect. 4.5). A TIRP-prefix's instance was evaluated at each time stamp as long as the model does not make a decision on whether the TIRP will be unfolded or until the end of the entity's data. In case of the entity's data were over, the decision was determined as the TIRP will not be unfolded.

The CSP, AHE, and DBT datasets were abstracted with eleven combinations of temporal abstraction methods (KB, GRAD, EWD, EFD, and SAX) and the number of symbols (2, 3, 4, and KB), in which the TIRPs were discovered for each combination. The EFIF dataset was not abstracted using KB and resulted in ten combinations of temporal abstraction methods (GRAD, EWD, EFD, and SAX) and the number of symbols (2, 3, and 4). All five continuous prediction models (SCPM, FCPM, RF, ANN, and RNN) were evaluated on the TIRP-prefixes' detected instances. The decisions were made based on a prediction decision threshold and the $\tau$, in which the metrics (Sect. 5.3) were computed based on the threshold's different configurations.

The prediction decision threshold was used with values between 0 to 1 with an increment of 0.1, and the $\tau$ was varied using 0, 1, 2, and 3 time units.

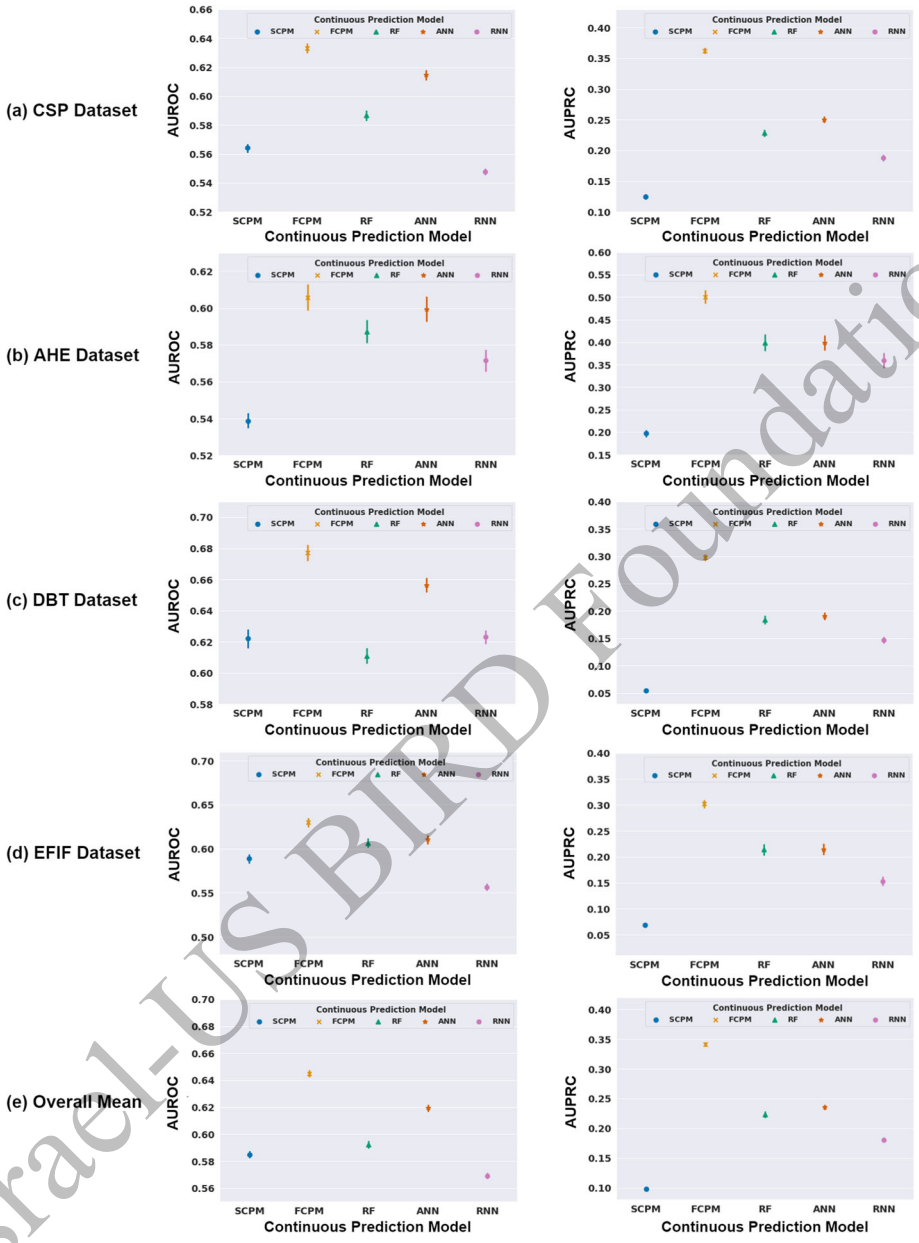



**Fig. 12** FCPM performed with much better AUROC and AUPRC than the baseline models for all datasets. Out of all baseline models, ANN performed with better AUROC and AUPRC. In contrast, the AUPRC performances of SCPM were lower than other models

### 5.3 Evaluation metrics

To evaluate the CPMs' performance, a receiver operating characteristics (ROC) curve was calculated and the corresponding area under the curve (AUROC). However, since the imbalanced ratio differed between the patterns, we also used a precision-recall (PR) curve and computed the area under the PR curve (AUPRC), in which the PR curve gives a more informative picture of an algorithm's performance for highly skewed data [54]. In contrast to the AUROC, in which a random classifier gets a score of 0.5, the AUPRC of a random estimator calculates as the number of *minority class* examples divided by the total number of examples [54, 55]. In this study, the prediction decision thresholds described in Sect. 4.5 varied from 0 to 1 to create the ROC and PR curves.

In the following section (Sect. 6), the mean AUROC and AUPRC results of the CPMs include confidence intervals of 95%.

## 6 Results

The results are based on 2,392 10-fold cross-validation runs on 1,094 TIRPs for the CSP dataset, 158 TIRPs for the AHE dataset, 766 TIRPs for the DBT dataset, and 374 TIRPs for the EFIF dataset.

To answer the research question RQ1, regarding which CPM performs better for a TIRP's completion, we tested the overall performance of the five models: SCPM, FCPM, RF, ANN, and RNN while using the early warning strategies. Figure 12 presents the mean AUROC and AUPRC results for CPMs, averaging the results of the different patterns. Each point on the graph represents the mean performance results of the CPM in predicting the TIRPs' completions with different values of $\tau$.

Figure 12 shows that RF and ANN performed similarly in terms of AUPRC. However, ANN was better in terms of AUROC. In addition, in terms of AUPRC, SCPM performed worst for all datasets, and RNN was the second-worst model. FCPM performed with better AUROC and AUPRC than the baseline models for all datasets. Out of all baseline models, ANN performed with better AUROC and AUPRC.

To answer the research question RQ2 regarding the best value of $\tau$ for a TIRP's completion, we evaluated the performances of the CPMs while using the $\tau$ parameter with 0, 1, 2, or 3 time units. Figure 13 presents the mean AUROC and AUPRC results for each CPM and $\tau$. Each point on the graph represents the mean performance results of the CPM in predicting the TIRPs' completions.

Figure 13 shows that the $\tau$ of two time units performed best in terms of AUROC for all CPMs in the CSP, AHE, and EFIF datasets. For the DBT dataset, a lower $\tau$ performed better in AUROC for most of the CPMs. In terms of AUPRC, the different time delays overall performed similarly. FCPM performed better than other baseline models for each $\tau$. In addition, the AUPRC performances of SCPM were lower than other models. In terms of AUROC, ANN performed better than the other baseline models, and RNN and SCPM performed the worst.

To answer the research question RQ1, regarding how early each CPM can predict a TIRP's completion, we evaluated the performances of the CPMs while using the $\tau$ parameter with 0, 1, 2, or 3 time units. Figure 14 presents the instances' revealed time portions for the TIRP's completion, in which only the true positive cases (i.e., when the TIRP's completions

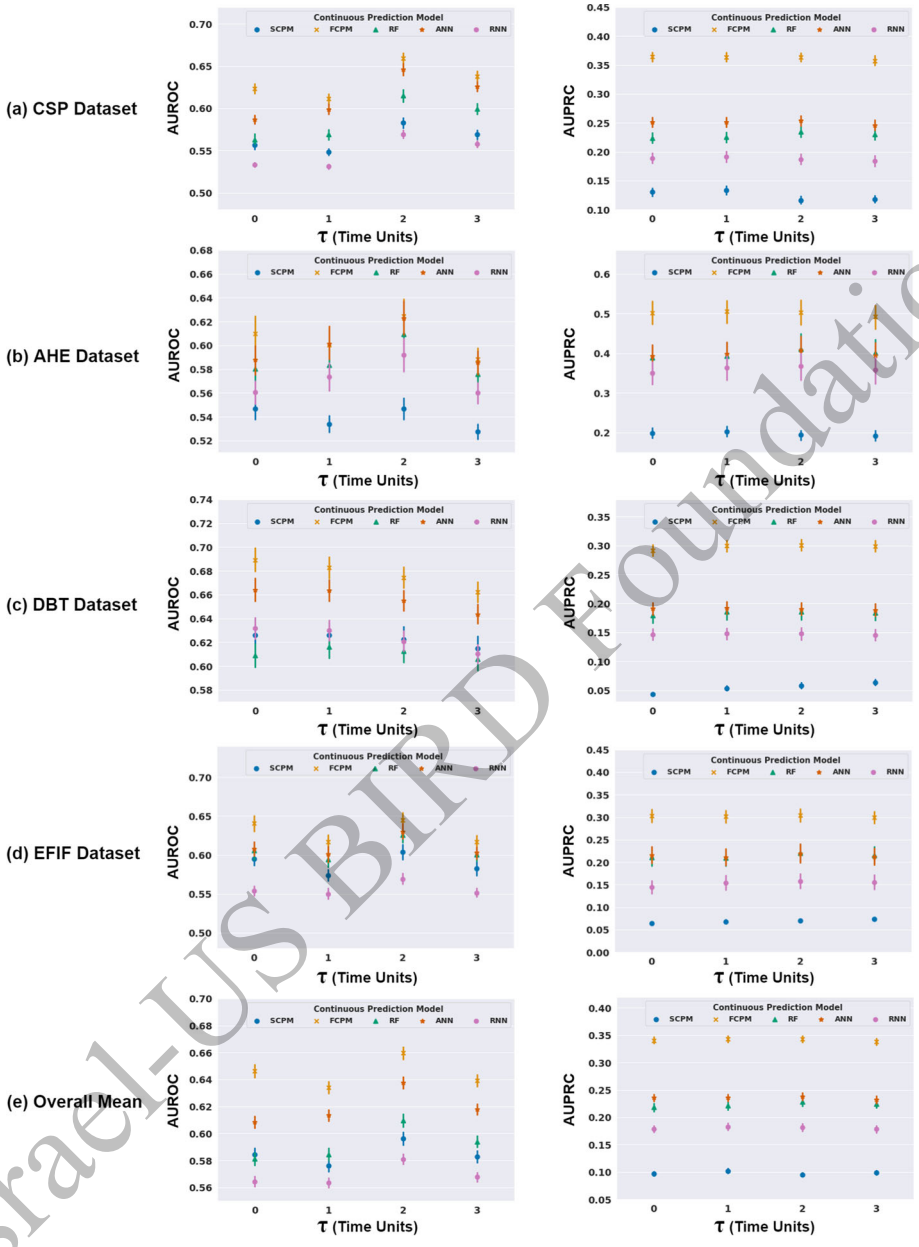



**Fig. 13** In terms of AUPRC, the different time delays overall performed similarly, and FCPM performed better than other baseline models for each $\tau$. In addition, the AUPRC performances of SCPM were lower than other models

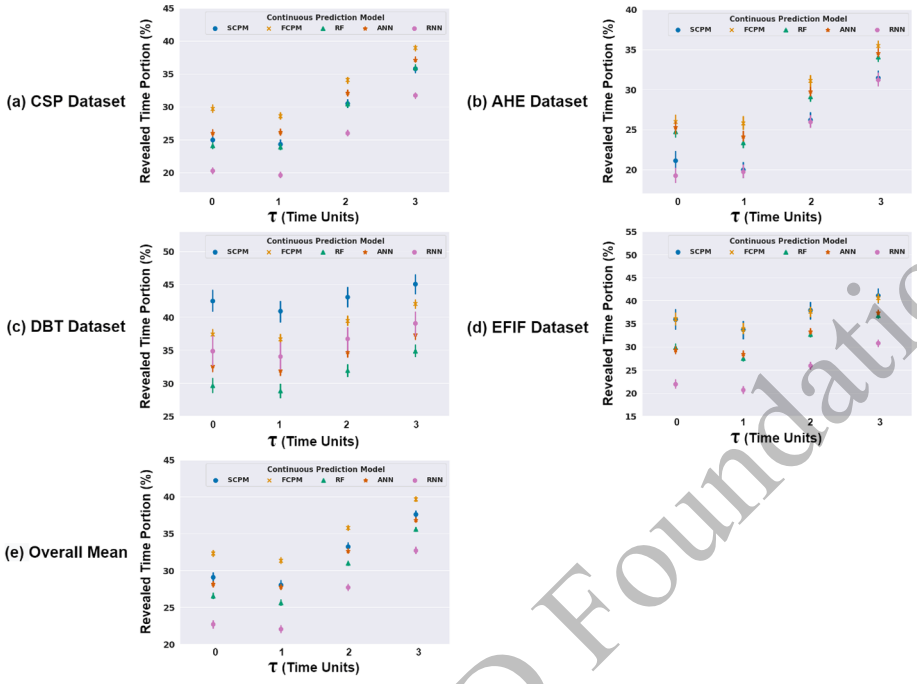



**Fig. 14** For TIRP's completions' correctly predicted cases, the CPMs provided earlier predictions when the $\tau$ was lower. In addition, RNN provided the earliest predictions, and FCPM provided the latest predictions

were correctly predicted) were considered for each $\tau$ result. Each point in the results' graphs represents the average result for the patterns while varying the prediction decision thresholds.

Figure 14 shows that when the TIRP's completions were correctly predicted, the models provided earlier predictions when the $\tau$ was lower, as expected. For the true positive cases using the CSP, AHE, and EFIF datasets, RNN provided the earliest predictions, while in the first two datasets, FCPM provided the latest predictions. For the DBT dataset, RF provided the earliest predictions while SCPM provided the latest predictions. However, there is a trade-off between the prediction performance and the earliness prediction of the TIRP's completion (Fig. 15).

To analyze the trade-off between the prediction performance and how early each CPM can predict a TIRP's completion, we evaluated the performance of the CPMs while using the $\tau$ parameter with 0, 1, 2, or 3 time units. Figure 15 presents the instances' revealed time portions for the TIRP's completion.

Each point in the results' graphs represents the average result for the patterns for each CPM and $\tau$, in which the revealed time portions were resulted by varying the prediction decision thresholds for each $\tau$ for the true positive cases.

Figure 15 shows that using the CSP, AHE, and EFIF datasets, RNN provided the earliest predictions. However, its prediction performances were relatively poor. Also, FCPM provided the latest but most accurate predictions. In all datasets, there is a trade-off between the prediction performance and their earliness, in which more accurate CPMs also need more time for making the decisions. These results strengthen the results presented in the preliminary analysis (Appendix E.2), in which, as long as time goes by for each instance, the CPMs provided more accurate predictions.

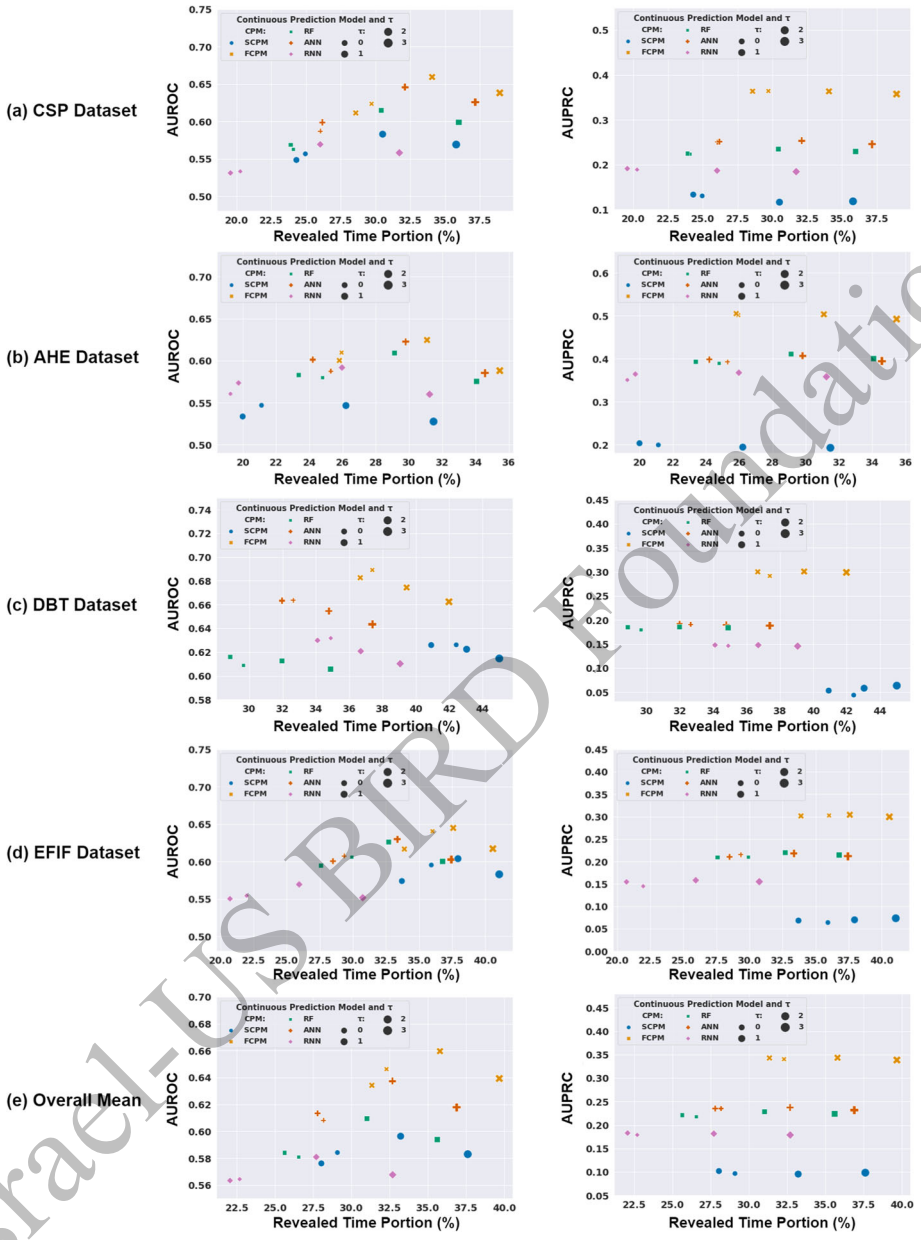



**Fig. 15** There is a trade-off between the prediction performance and their earliness, in which more accurate CPMs also need more time for making the decisions

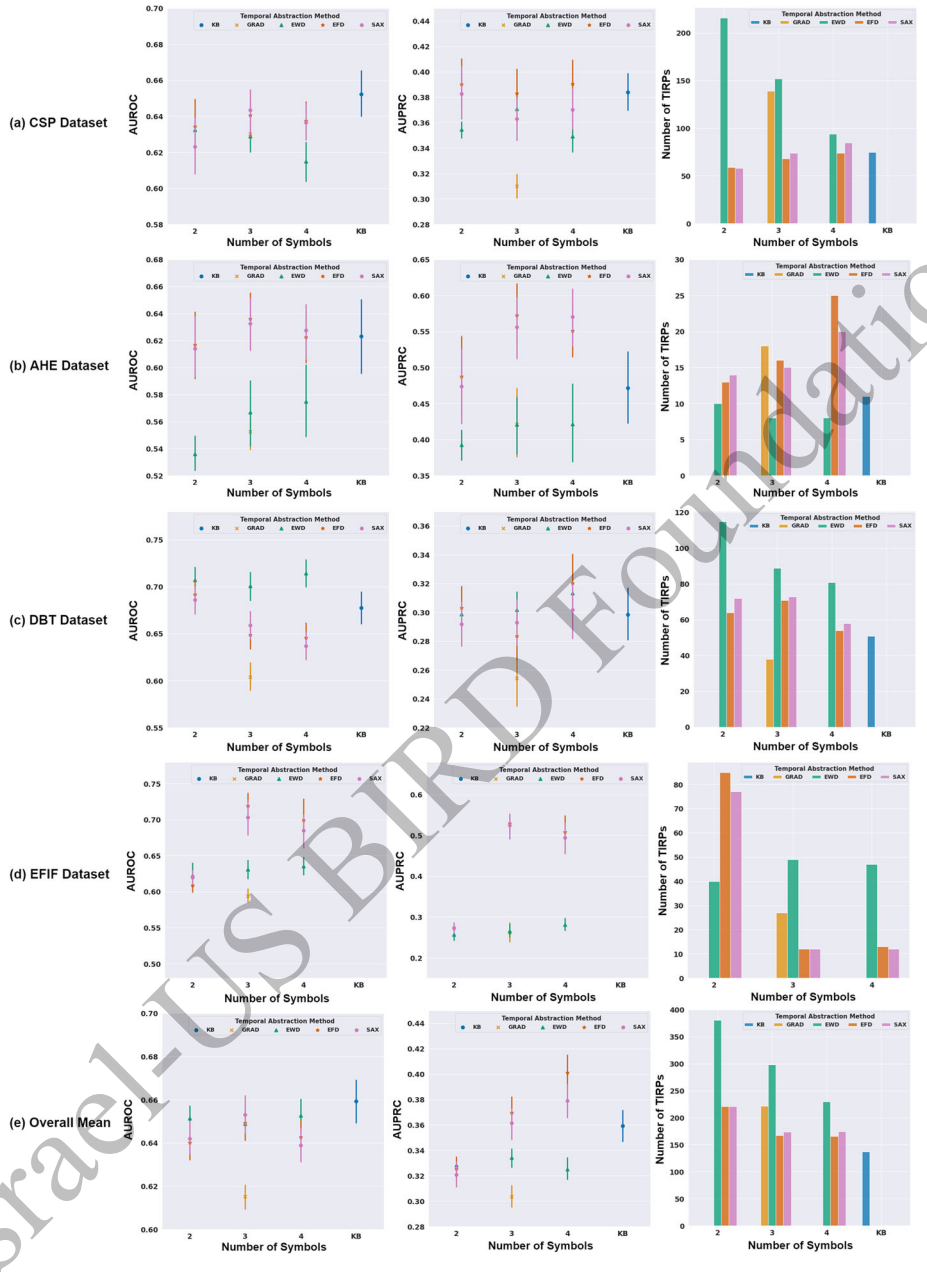





**Fig. 16** There is no significantly better temporal abstraction method and a number of symbols for the task of TIRP's completion

To answer research question RQ3, we evaluated the performances of FCPM while predicting TIRPs' completions of patterns that were discovered from data abstracted using different temporal abstractions methods and the numbers of symbols. The tested temporal abstraction methods were: KB, GRAD, EFD, EWD, and SAX, and the numbers of symbols per variable were: two, three, four, and a varied number of symbols for KB. Note that GRAD was evaluated with only three symbols. Figure 16 presents the number of patterns per temporal abstraction method and the number of symbols. The mean AUROC and AUPRC for each tested combination of temporal abstraction method and number of symbols included the different patterns with values of $\tau$.

Figure 16 shows that CSP and DBT datasets that were abstracted with EWD resulted in more patterns that end with the target event, while EFD and SAX resulted in more patterns in the AHE dataset. Also, it can be shown that abstracting the datasets with two symbols per variable resulted in more patterns that end with the target event. The FCPM performed worst with GRAD in the CSP, DBT, and EFIF datasets in terms of AUPRC. In terms of AUROC, KB performed best for the CSP dataset and EWD best for EWD. Also, SAX and KB performed best for the AHE dataset. In the AHE and EFIF datasets, EWD and GRAD performed worst. Overall, there is no significantly better temporal abstraction method and a number of symbols for the task of TIRP's completion.

In summary, FCPM performed with better AUROC and AUPRC than the baseline models for all datasets. Out of all baseline models, ANN performed with better AUROC and AUPRC. In contrast, the AUPRC performances of SCPM were lower than other models. In addition, in terms of AUPRC, the different time delays overall performed similarly, and FCPM performed better than other baseline models for each $\tau$. In addition, the AUPRC performances of SCPM were lower than other models. In terms of AUROC, ANN performed better than the other baseline models, and RNN and SCPM performed the worst. For TIRP's completions' correctly predicted cases, the CPMs provided earlier predictions when the $\tau$ was lower. In addition, RNN provided the earliest predictions, and FCPM provided the latest predictions. However, there is a trade-off between the prediction performance and their earliness, in which more accurate CPMs also need more time for making the decisions.

## 7 Discussion and conclusions

This paper studied the continuous prediction of a TIRP's completion for the first time. Continuously predicting a TIRP's completion, based on partial information, can be beneficial in real-life data in multiple domains and applications, such as predictive process monitoring [14–16] or when a temporal pattern ends with an event of interest, such as the recovery of a patient, or an undesirable clinical event, such as death or a medical complication. For example, an acute blood pressure elevation [12], is used in this study, in which the data were extracted from a real ICU database. Moreover, this method can be useful and effective in heterogeneous multivariate temporal data, in which many methods fail to incorporate all types of temporal variables due to their heterogeneous sampling forms. As we demonstrate in this study, temporal abstraction transforms the various variables into STIs representation, enabling applying the TIRP's completion to predict whether target events will occur in the data, based on known patterns ending with them.

In this study, the continuous TIRP's completion methodology was evaluated on TIRPs, ending with a target event of interest. However, the main challenge in performing such a prediction lies in the *unfinished coinciding STIs* (Sect. 3.3), which do not allow determining

the "transition probability" among the TIRP's evolution process and introduce an uncertainty in the evolving temporal relations. For that, we introduced the TIRP-prefixes representation to overcome those challenges.

We proposed two continuous prediction models (CPMs) for a TIRP's completion, in which the SCPM ignored the time duration between the *tieps*. The FCPM, unlike the SCPM, incorporates the time duration between the TIRP-prefix's consecutive *tieps* and thus allows dynamic change in the TIRP's completion estimation, even during the occurrence of the STIs. Each CPM is based on a single TIRP since the study aims to learn the TIRP's completion prediction problem and tackle fundamental challenges in this domain problem. Finally, the method includes early warning strategies defined by a threshold and a parameter $\tau$ that enables conservative decision making.

A rigorous evaluation of four real-life datasets was performed. The FCPM performed with better AUROC and AUPRC than the SCPM and the baseline models for all datasets. While evaluating the CPMs' ability to make decisions over time, using the early warning strategies, the FCPM was significantly better than the baseline models, with an average AUROC gap of 2% and an AUPRC gap of 8% (Fig. 12). Out of all baseline models, the ANN performed with better AUROC and AUPRC. In contrast, the AUPRC performances of the SCPM were lower than other models. However, while analyzing the CPMs at the instances' different portions over time (Appendix E), the AUPRC of the SCPM performed worse at making relatively early decisions but improved drastically as time progressed.

Additionally, in terms of AUPRC, the different time delays overall performed similarly, and the FCPM performed better than the SCPM and the other baseline models for each $\tau$. For TIRP's completions' correctly predicted cases, the CPMs provided earlier predictions when the $\tau$ was lower. However, there is a trade-off between the prediction performance and their earliness, in which more accurate CPMs also need more time, and observations through the time, for making a better estimation. This trade-off results since new data are revealed over time (i.e., the TIRP-prefix's *tieps* and the duration between them), which increases the CPM's prediction performance.

To summarize, this paper presents a new method for predicting the completion of TIRPs in real-world data, utilizing two CPMs—the SCPM and the FCPM. The FCPM consistently outperformed the SCPM and baseline models across all datasets. However, there is a trade-off between earliness and accuracy since the starting and ending times of new TIRPs are gradually revealed over time. This proposed method has potential applications in domains such as healthcare and predictive process monitoring. Additionally, our experiments demonstrated that the average prediction accuracy for TIRP completion is around 65% AUROC and 35% AUPRC, but this can be further improved by using multiple patterns simultaneously or prioritizing more predictive patterns.

One of the limitations of our study was that each TIRP was evaluated separately and, as a result, included different instances, which makes the evaluation more difficult since the imbalanced ratio was different (Appendix C). In addition, evaluating the TIRP's completion earliness included only the true positive (TP) cases (i.e., predicted that the TIRP will be unfolded to completion and corrected), in which the number of TP cases was different between the CPMs and for the different values of decision thresholds.

In the future, we intend to expand the proposed methods to continuously consider multiple frequent TIRPs, which can be used for continuous prediction of real-life tasks such as event prediction of common complications in critically ill patients [12, 18] or recovery events. A model based on multiple TIRPs, ending with the same target event, is expected to be more generalizing and accurate in predicting the target event. However, the process of discovering frequent TIRPs for TIRPs' completion can result in many unpredictive patterns, which we

would like to investigate in our future work further. In addition, we intend to consider the continuous change of the statistical properties of the time durations' distributions by using incremental learning, in which new input data will be used continuously to update the existing model's knowledge.

To sum up, this study proposed a method for continuously predicting the completion of TIRPs, which has potential uses in various domains and applications. The ability to predict target events based on known patterns can be particularly useful in domains such as healthcare and predictive process monitoring. The proposed method is effective in handling heterogeneous multivariate temporal data and includes early warning strategies, enabling supportive decision making. Applying this method to multiple TIRPs simultaneously has the potential to improve outcomes for patients and increase efficiency in various industries.

**Acknowledgements** Nevo Itzhak was funded by the Kreitman School of Advanced Graduate Studies and the Israeli Ministry of Science and Technology Jabotinsky scholarship grant #3-16643.

**Author Contributions** N.I., S.J., and R.M. designed research. N.I. and R.M. performed research. N.I. analyzed data. N.I., S.J., and R.M. wrote the paper.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## Appendix A Efficient TIRP's candidates generator

We introduce an efficient TIRP's candidates generator (Algorithm 3), which exploits the transitivity property to generate all possible TIRPs that can evolve from a given TIRP-prefix with multiple unfinished STIs. The algorithm works by iterating over all possible temporal relations between each pair of adjacent unfinished STIs, according to the lexicographical order, and inferring temporal relations between the remaining unfinished STI pairs using Allen's transition Table [26]. The relevant temporal relations that can eventually evolve given unfinished STIs $A^*$, $B^*$ and $C^*$ are presented in Table 1. Given the temporal relation that eventually evolved between STIs $A$ and $B$, $r(A,B)$, and the temporal relation between STIs $B$ and $C$, $r(B,C)$, can be used to infer the optional temporal relations between STIs $A$ and $C$, $r(A,C)$.

Algorithm 3 takes a TIRP-prefix as inputs and stores the set of possible TIRPs that can evolve from it in the variable *fnlTIRPCnddts*. The given TIRP-prefix might include both finished STIs ($\hat{IS}_f$) or unfinished STIs ($\hat{IS}_*$). Only the temporal relations among pairs of

**Table 1** The Allen's transition table that is relevant for the unfinished coinciding STIs challenge

| r(A,B) \ r(B,C) | Overlaps (o) | Finished-by (fi) | Contains (c) | Equals (=) | Starts (s) |
|---|---|---|---|---|---|
| Overlaps (o) | *o* | *o* | *o, fi, c* | *o* | *o* |
| Finished-by (fi) | *o* | *fi* | *c* | *fi* | *o* |
| Contains (c) | *o, fi, c* | *c* | *c* | *c* | *o, fi, c* |
| Equals (=) | *o* | *fi* | *c* | = | *s* |
| Starts (s) | *o* | *o* | *o, fi, c* | *s* | *s* |

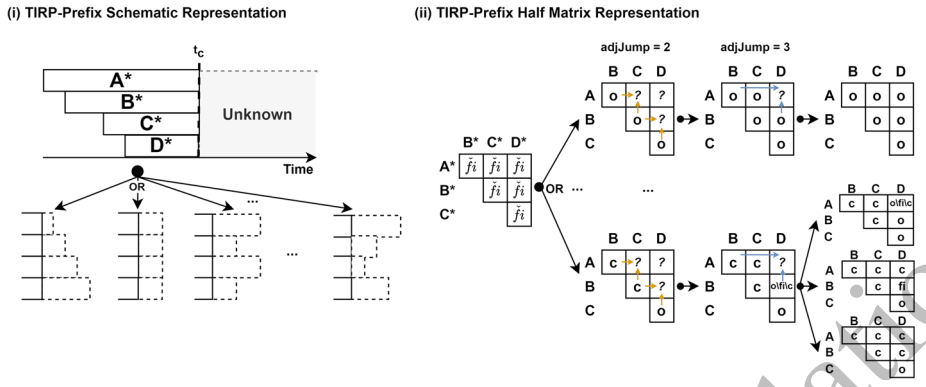



**Fig. 17** A TIRP-prefix including unfinished STIs $A^*$, $B^*$, $C^*$, and $D^*$, where $A^*+ < B^*+ < C^*+ < D^*+$ and all possible TIRPs that it can evolve into, presented in two ways: (i) TIRP-prefix schematic and (ii) TIRP-prefix half matrix representation

unfinished STIs are unknown and need to be determined. First, *fnlTIRPCnddts* is initialized to the empty set (line 1), and a variable *unfSTIsLen* is set to the number of unfinished STIs in the TIRP-prefix (line 2). Then, the function *initGenAdjacentUnfSTIs* enumerates all possible temporal relations between all pairs of adjacent (according to the lexicographical order) unfinished STIs of the TIRP-prefix. Thus, for $\check{k}$ unfinished STIs, the function outputs $3^{\check{k}-1}$ possible candidates, stored in the variable *initTIRPCanddts* (line 3). In lines 4–6, the algorithm iterates over *initTIRPCanddts*. Each candidate $c$ is expanded using the function *expandTIRPCand* (Algorithm 4), which returns all TIRP candidates derived from $c$ using Allen's transition table (line 5). They are added to *fnlTIRPCnddts*, which is finally returned by the algorithm.

**Algorithm 3** Efficient TIRP's Candidates Generator

**Input**: TIRPPrefix - a TIRP-prefix.
**Output**: fnlTIRPCnddts - set of TIRPs that can evolve from the TIRP-prefix.

1: fnlTIRPCnddts ← ∅
2: unfSTIsLen ← length(TIRPPrefix.$\hat{IS}_*$) ▷ number of unfinished STIs
3: initTIRPCanddts ← initGenAdjacentUnfSTIs(TIRPPrefix.$\hat{IS}_*$) ▷ get candidates
4: **for each** c **in** initTIRPCanddts **do**
5: extTIRPCand = expandTIRPCand(c, unfSTIsLen, adjJump=2, i=0)
6: fnlTIRPCnddts = fnlTIRPCnddts ∪ extTIRPCand
7: **return** *fnlTIRPCnddts*

The function *expandTIRPCand* takes a parameter *adjJump* which specifies the "adjacency jump," i.e., the distance between the analyzed adjacent unfinished STIs. For example, in Fig. 17, given the temporal relations between the adjacent unfinished STIs: *A overlaps B*, *B overlaps C*, and *C overlaps* D, and while considering the second-adjacent unfinished STIs (*adjJump*=2), $r(A, C)$ is must be *overlaps* based on $r(A, B)$ and $r(B, C)$. Similarly, $r(B, D)$ must be *overlaps* based on the $r(B, C)$ and $r(C, D)$. Then, the recursive procedure proceeds by inferring all the temporal relations among $c$'s *adjJump*+1 adjacent unfinished STIs. For example, while considering the third-adjacent unfinished STIs in Fig. 17, $r(A, D)$ can be inferred from $r(A, B)$ and $r(B, D)$.

Algorithm 4 presents the recursive *expandTIRPCand* function. The function takes as arguments the current TIRP candidate to expand ($c$), the number of TIRP-prefix's unfinished STIs (*unfSTIsLen*), the adjacency jump between unfinished STIs (*adjJump*), and an index $i$ of the earliest unfinished STI to consider. The function returns a set of $c$'s expanded TIRPs candidates. In lines 2–3, a function *getRel* is called, which returns the temporal relation between two given unfinished STIs. Note that the temporal relation between the $i$-th and $i+1$-th unfinished STIs (stored in *fstRel*) is known since it was determined during the enumeration step of Algorithm 3. Also, the temporal relation between the $i+1$-th and $i+adjJump$-th unfinished STIs (stored in *scdRel*) is known since it has been determined earlier in the recursive procedure. Based on *fstRel* and *scdRel*, the temporal relations among the $i$-th and $i+adjJump$-th unfinished STIs are inferred using Allen's transition table and stored in *inferredRels* (line 4).

**Algorithm 4** Expand TIRP's Candidates (*expandTIRPCand* function)

**Input**: $c$ - TIRP candidate to expand; *unfSTIsLen* - number of TIRP-prefix's unfinished STIs; *adjJump* - the adjacency's jump among the unfinished STIs; $i$ - index of the earliest unfinished STI.
**Output**: fnlCands - set of expanded TIRPs candidates.

```
1: fnlCands ← ∅
2: fstRel = getRel(c, i, i+1)                          ▷ get temporal relation between two given unfinished STIs
3: scdRel = getRel(c, i+1, i+adjJump)
4: inferredRels = transitionTable(fstRel, scdRel)      ▷ get only possible temporal relations
5: for each rel in inferredRels do
6:     extC = updateRel(c, i, i+adjJump, rel)
7:     if i+1 < unfSTIsLen–adjJump then                ▷ more relations to infer
8:         fnlCands.union(expandTIRPCand(extC, unfSTIsLen, adjJump, i+1))
9:     else if adjJump+1 < unfSTIsLen then             ▷ more relations to infer
10:        fnlCands.union(expandTIRPCand(extC, unfSTIsLen, adjJump+1, i))
11:    else
12:        fnlCands.union(extC)
13: return fnlCands
```

Then, the algorithm iterates over the *inferredRels* (lines 5–12) with the current relation stored in the variable *rel*, and in each iteration executes the following steps: (a) updates $c$'s temporal relation between the $i$-th and $i+adjJump$-th unfinished STIs, which is stored in *extC* (line 6); (b) if there are more relations between *adjJump*-adjacent STIs to infer, calls itself recursively with *extC* and *i+1* adding the result to *fnlCnds* (lines 7–8); (c) otherwise, if there are more relations between *adjJump+1*-adjacent STIs to infer, calls itself recursively with *extC* and *adjJump+1* adding the result to *fnlCnds* (lines 9–10); (d) otherwise, adds *extC* to *fnlCnds* (lines 11–12). Lastly, *fnlCands* is returned.

Given a TIRP-prefix with STI series $IS$ of size $k$, in which there are $\check{k}$ unfinished STIs, the overall time complexity of creating the TIRP's candidates, using Algorithms 3 and 4, is bounded in the worst case by the following expression: $O(3^{(\check{k}^2-\check{k})/2})$. The base of the exponent in the upper bound (i.e., three) represents the number of possible temporal relations between each pair of unfinished STIs. The formula $(\check{k}^2-\check{k})/2$ follows from the binomial coefficient $\binom{\check{k}}{2} = \check{k}(\check{k}-1)/2 = (\check{k}^2-\check{k})/2$, where two stands for pairs of temporal relations and $\check{k}$ is the number of unfinished STIs. However, this worst-case, upper-bound complexity expression is far from the practical situation for the algorithms. The reduction is achieved

by using the transitivity of temporal relations. The overall number of generated candidates is smaller than the naive generation, which does not exploit the transitivity property.

## Appendix B Cutoffs used for the knowledge-based abstraction

The knowledge-based abstraction was performed using cutoffs defined by a domain expert, in which the states for each dataset were as follows:

- **CSP dataset**:
  - MAP [mmHg]: $\leq$60, (60,90], >90
  - CVP [mmHg]: $\leq$5, (5,17], >17
  - FiO2 [%]: $\leq$41, (41,60], >60
  - HR [bpm]: $\leq$60, (60,110], >110
  - PEEP [cmH2O]: $\leq$10, >10
  - TMP [°C]: $\leq$35, (35,38.5], >38.5
  - BE [mEq/L]: $\leq$-6, (-6,-3], >-3)
  - CI: $[L/min/m^2]$ $\leq$2.5, >2.5
  - Glucose [mmol/L]: $\leq$2.5, (2.5,10] >10
  - CKMB [%]: $\leq$25, (25,50], >50
- **AHE dataset**:
  - HR [bpm]: $\leq$50, (50, 100], >100
  - RESP [breath/min]: $\leq$7, (7,20], >20
  - SpO2 [%]: $\leq$88, (88,100], >100
  - SABP [mmHg]: $\leq$90, (90,140], >140
- **DBT dataset**:
  - Blood Glucose [mg/dL]: $\leq$100, (100,125], (126,200], >200
  - HbA1C [%]: $\leq$7, (7,9], (9,10.5], >10.5
  - LDL Cholesterol [mg/dL]: $\leq$100, (100,130], (130,160], >160
  - Creatinine [mg/dL]: $\leq$1, (1,1.5], (1.5,2.5], (2.5,4], >4
  - Albumin [g/dL]: $\leq$3.5, >3.5

The cutoffs for the EFIF dataset were not applicable.

## Appendix C imbalance ratio

The TIRPs' instances imbalance ratio was defined as the number of instances of a complete TIRP divided by the total number of instances. Figure 18 presents the number of TIRPs' instances imbalance ratio and the performance prediction while using the early warning strategies for each TIRPs' instances imbalance ratio and values of $\tau$. As can be shown in Fig. 18, the imbalance ratio was lower than 0.3 for most of the TIRPs. Overall, there was a positive correlation between the AUPRC performances and the imbalance ratio for all datasets. In contrast, there was no significant correlation between the imbalance ratio and the AUROC.

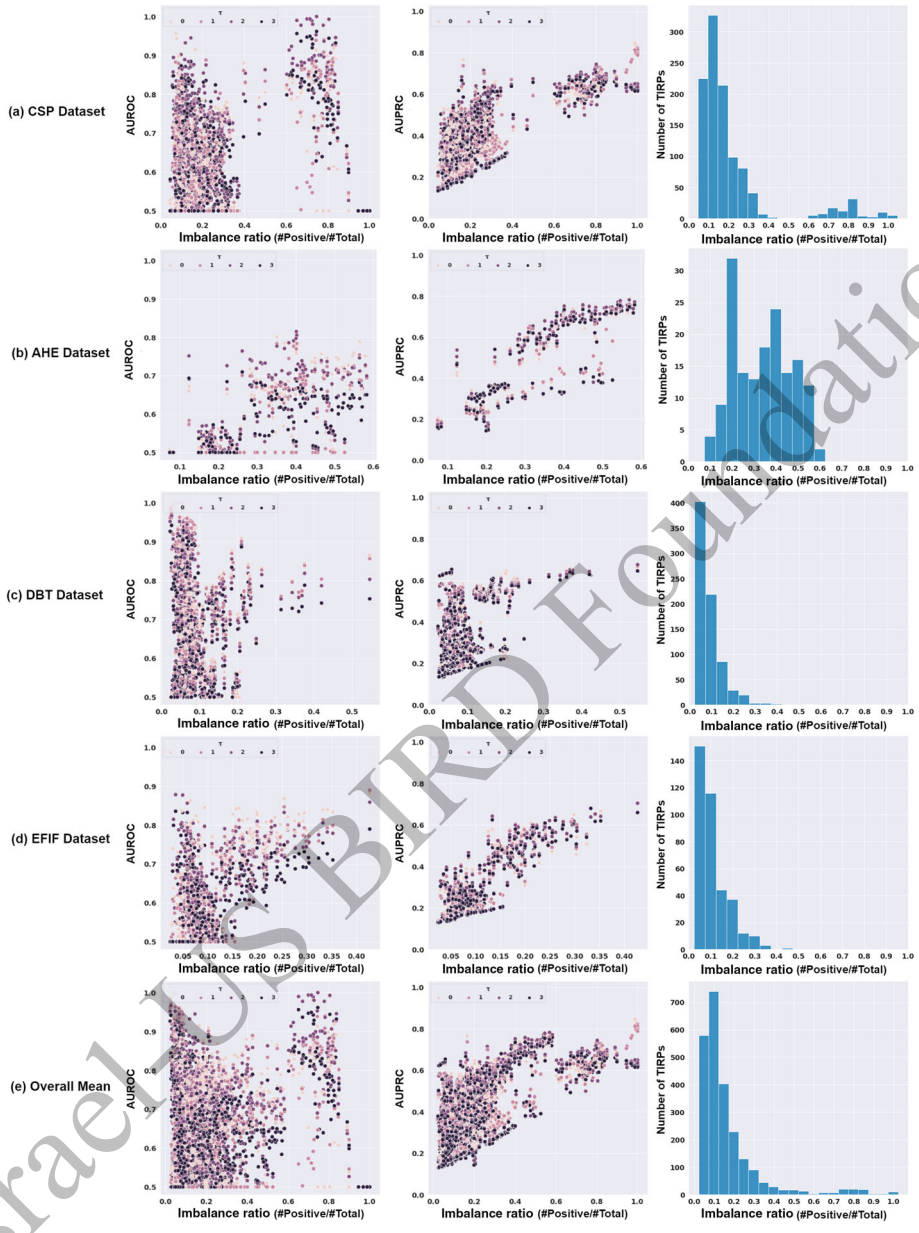



**Fig. 18** The imbalance ratio was lower than 0.3 for most of the TIRPs, and there was a positive correlation between the AUPRC performances and the imbalance ratio for all datasets

## Appendix D Parameters of baseline models

The parameters of each model are selected after testing the performance of each combination (not in a greedy comparison approach), and here we describe the parameters that performed best.

Random forest (RF) [51] classifier utilizes an ensemble learning approach, a technique that combines the decisions from multiple models. We used 100 trees in the forest for the RF model and a maximum depth of 3 for a tree, in which the combinations of 30, 50, 100, and 200 trees in the forest and depths of 2, 3, 5, or 10 were tested. Bootstrap was used when building trees and out-of-bag samples to estimate the generalization accuracy. RF was implemented with Python 3.6 Scikit-Learn (https://scikit-learn.org) version 0.22.1.

The artificial neural network (ANN) [52] is comprised of neuron layers, which contain an input layer, hidden layers, and an output layer. In this paper, the fully connected ANN architecture was used, in which all neurons in one layer are connected to all neurons in the next layer. We used ANN with two hidden layers with 50 neurons for each hidden layer and with the activation function ReLU [56]. We trained all models using a maximum epoch of 20, a batch size of 16, and a learning rate of 0.001 with gradually decreasing. A validation set was used to measure the generalization error by randomly taking 20 percent of all training data. We used early stopping on the validation set, in which lower than a change of 0.001 was not considered an improvement for the loss. For ANN, combinations of one, two, three, or five hidden layers with 20, 50, or 100 neurons and batch sizes of 16, 32, or 64 were tested. ANN was implemented with Python 3.6 Scikit-Learn (https://scikit-learn.org) version 0.22.1.

The sequential deep learning recurrent neural network (RNN) [53] is designed to learn the data dependencies in a sequence. We used RNN with five hidden units per layer and a recurrent dropout of 0.2 and an activation function, ReLU [56]. We trained all models using a maximum epoch of 20, a batch size of 16, and a learning rate of 0.001 with a gradually decreasing. A validation set was used to measure the generalization error by randomly taking 20 percent of all training data. We used early stopping on the validation set, in which lower than a change of 0.001 for more than two epochs was not considered an improvement for the loss. For RNN, combinations of one, two, three, or five hidden units per layer, dropout of 0.2, 0.3, or 0.5, with batch sizes of 16, 32, or 64 were tested. RNN was implemented with Keras (https://keras.io) version 2.2.5.

For parameters we did not specify, we used the default.

## Appendix E Preliminary analysis

### E.1 TIRP's completion prediction relative to the end

This analysis's goal was to evaluate the prediction of a TIRP's completion at the instances' different portions over time using the CPMs (Sect. 5.2.1). The TIRP-prefixes' instances were evaluated at each time stamp until the end of the entity's data (i.e., patient), in which the decisions for the TIRP's completion were determined at different instances' revealed time portions. Then, the metrics (Sect. 5.3) were computed for all instances for each revealed time portion. The setup for this analysis was the same as described in Sect. 5.2.1. This preliminary analysis was excluded from the Evaluation and Results (Sects. 5–6) since it examines the TIRP's completion prediction in retrospect, in which the instances' different portions of decision points over time were selected based on the end of the entity's data.

The CSP, AHE, and DBT datasets were abstracted with eleven combinations of temporal abstraction methods (KB, GRAD, EWD, EFD, and SAX) and the number of symbols (2, 3, 4, and KB), in which the TIRPs were discovered for each combination. The EFIF dataset was not abstracted using KB and resulted in ten combinations of temporal abstraction methods (GRAD, EWD, EFD, and SAX) and the number of symbols (2, 3, and 4). All five continuous prediction models (SCPM, FCPM, RF, ANN, and RNN) were evaluated on the TIRP-prefixes' detected instances.

### E.2 Analysis results

The results are based on 2,392 10-fold cross-validation runs on 1,094 TIRPs for the CSP dataset, 158 TIRPs for the AHE dataset, 766 TIRPs for the DBT dataset, and 374 TIRPs for the EFIF dataset.

We first evaluated the overall performance of the five continuous prediction models: SCPM, FCPM, RF, ANN, and RNN on predicting a TIRP's completion at the instances' revealed portions of time. Figure 19 presents the mean AUROC, and AUPRC results over the instances' revealed portions of time, in which each point on the graph represents the mean performance results for the different TIRPs.

As expected, it can be seen from Fig. 19 that as long as time goes by for each instance, the continuous prediction models provided more accurate predictions, resulting in better AUROC and AUPRC performance over time. In all datasets, the FCPM performed best and the SCPM worst in terms of AUROC. However, in terms of AUPRC, the SCPM performed worst at making relatively early decisions regarding the instances of TIRP's completion and drastically improved as long as time goes. As a result, the SCPM performed with better AUPRC than the other baselines in all datasets at making a relatively late decision regarding the instances TIRP's completion. Also, for the DBT and EFIF datasets, even though the FCPM performed with better AUPRC than other baseline models at making relatively early decisions, SCPM performed better at making relatively late decisions. Overall, the FCPM performed better than the baseline models in terms of AUROC and AUPRC. The AUPRC performance of the SCPM was poor at making relatively early decisions but was improved drastically as long as time went by. As a result, the AUPRC performances of FCPM and SCPM were close at making relatively late decisions regarding a TIRP's completion.

The following results in this analysis are only involved in the FCPM results as we demonstrated its superiority.

Next, we evaluated the TIRP's completion at the instances' revealed portions of time by the FCPM for each temporal abstraction method (KB, GRAD, EWD, EFD, and SAX). Figure 20 presents the number of TIRPs per temporal abstraction method and the mean AUROC and AUPRC results over the instances' revealed portions of time. Each point on the chart represents the mean performance results of the FCPM in providing the completions predictions for the different TIRPs.

Figure 20 shows that the CSP, DBT, and EFIF datasets abstracted with EWD resulted in more TIRPs that end with the target event than other temporal abstraction methods. However, the AHE dataset that was abstracted with EFD resulted in more TIRPs that end with the target event. As can be seen from all datasets, the CPMs provided less accurate predictions for TIRPs discovered using GRAD. The prediction performances of EFD and SAX were quite similar, but SAX performed slightly better. Looking at the overall mean results, FCPM provided more accurate predictions for TIRPs discovered using KB and EWD. However, EWD performed much better than KB for the DBT dataset.

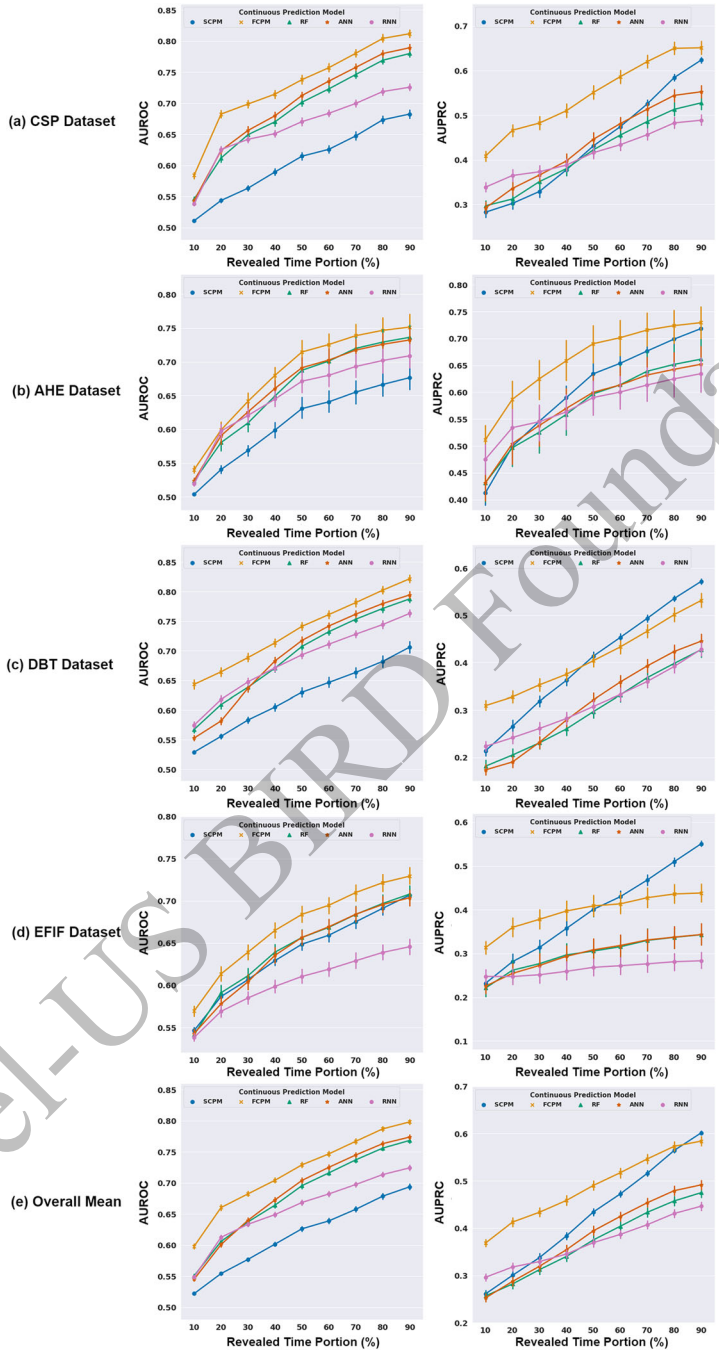



**Fig. 19** The FCPM performed better than the baseline models in terms of AUROC and AUPRC. The AUPRC performance of the SCPM was poor at making relatively early decisions but was improved drastically as long as time went by. As a result, the AUPRC performances of FCPM and SCPM were close at making relatively late decisions regarding a TIRP's completion

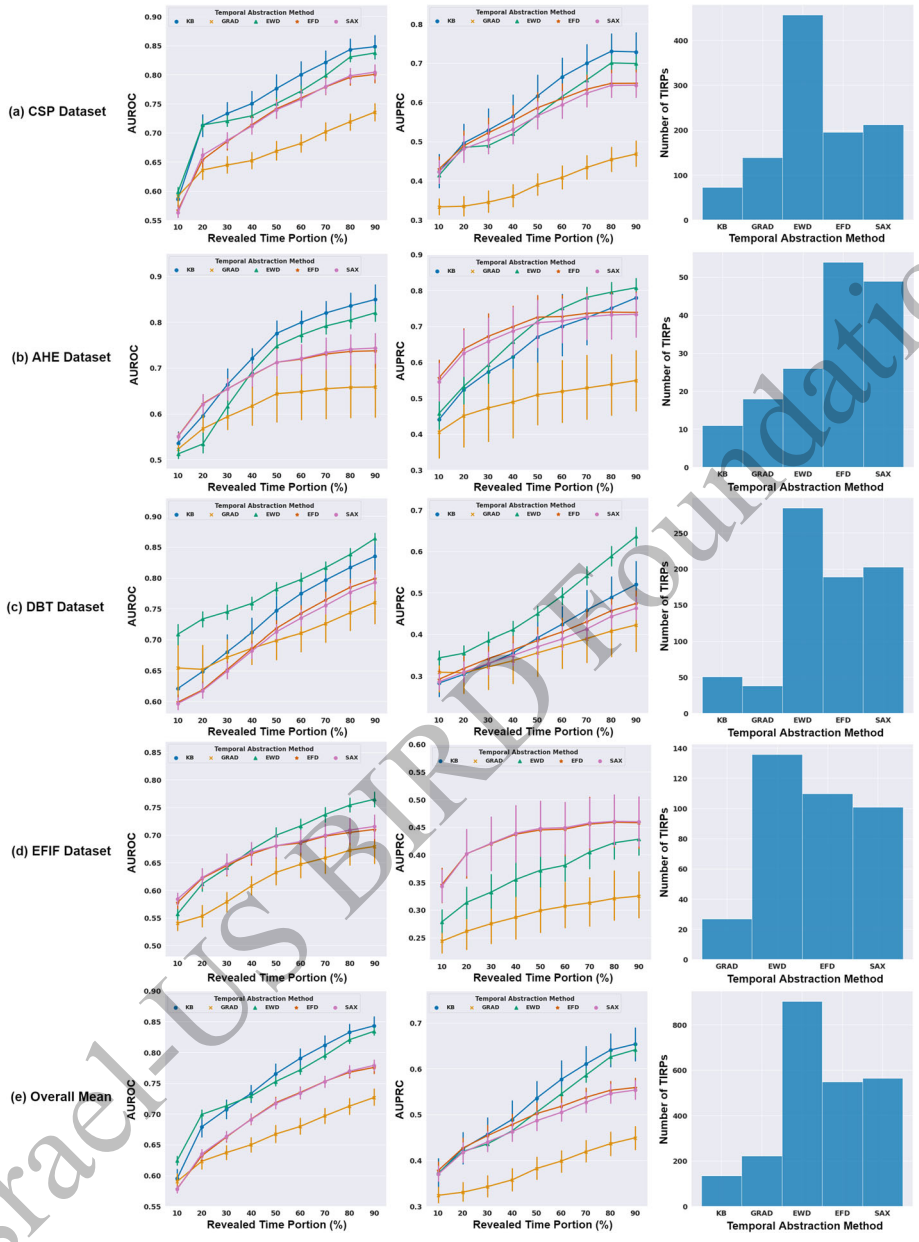



**Fig. 20** FCPM provided less accurate predictions for TIRPs discovered using GRAD and more accurate predictions for TIRPs discovered using KB and EWD

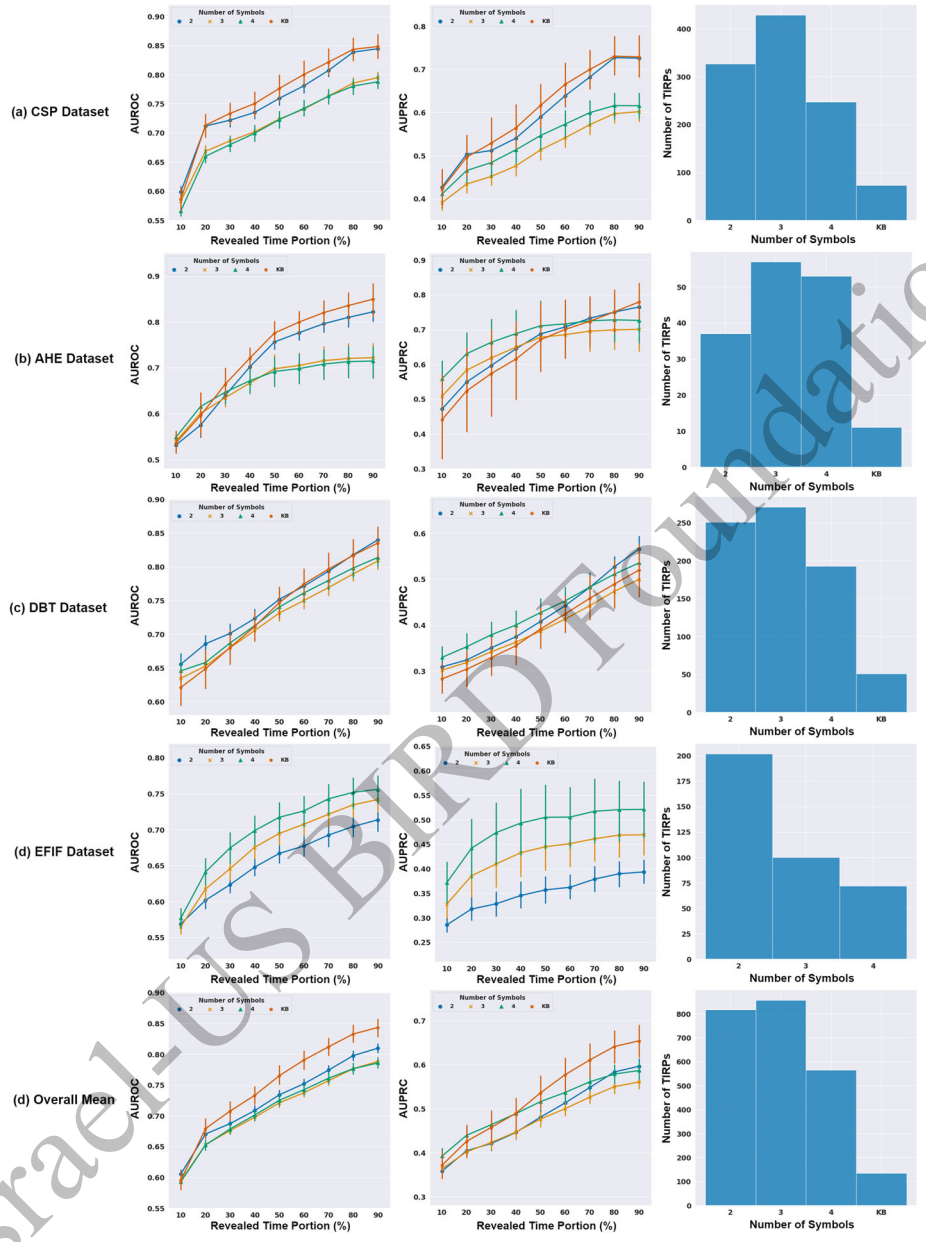



**Fig. 21** CPMs provided more accurate predictions over time for TIRPs discovered using two symbols per variable or KB than three and four symbols per variable

We also evaluated the TIRP's completion at the instances' revealed portions of time provided by the FCPM for each number of symbols (two, three, four, and a varied number of symbols for KB). Figure 21 presents the number of TIRPs per number of symbols and the mean AUROC and AUPRC results over the instances' revealed portions of time. Each point

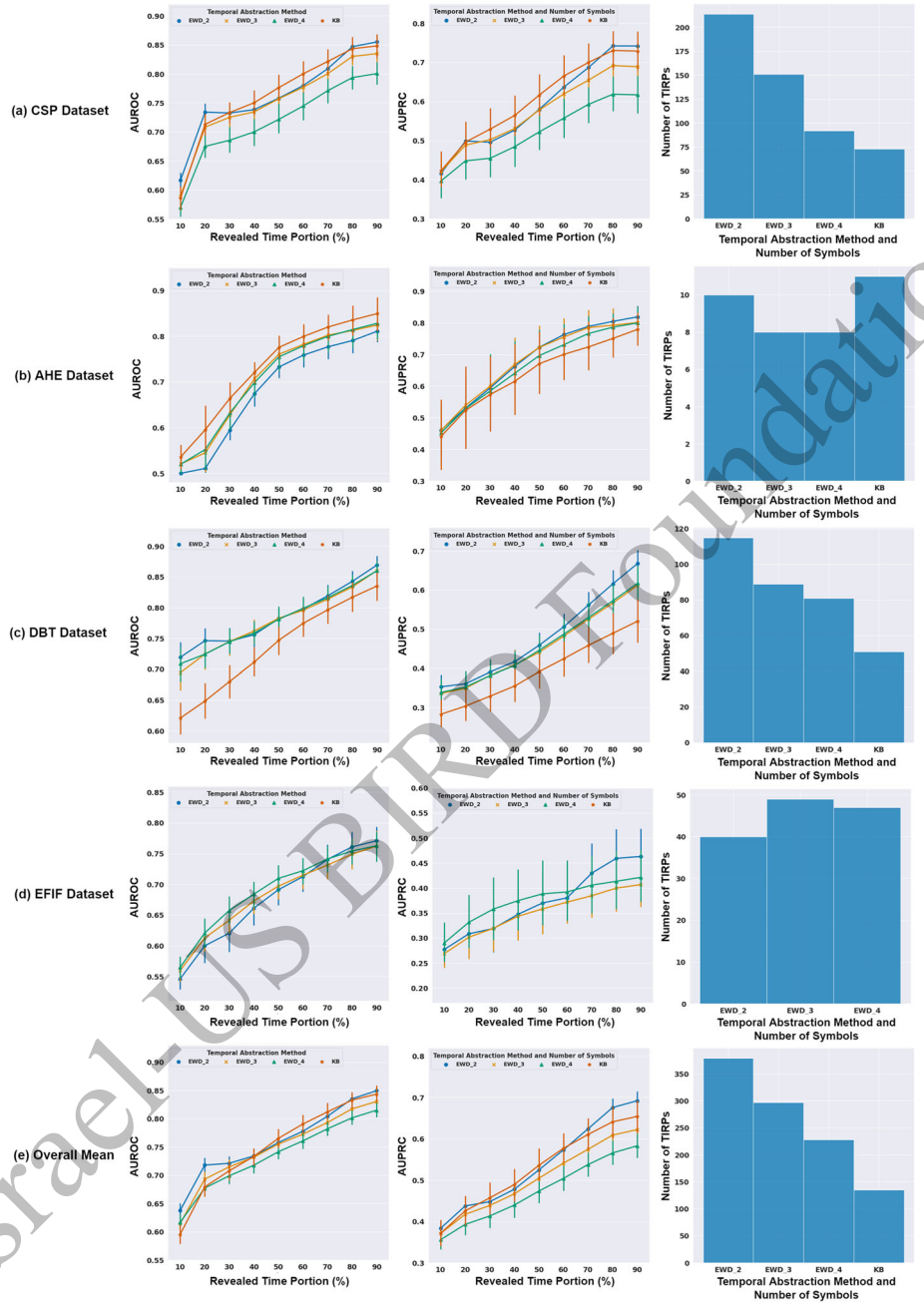





**Fig. 22** FCPM performed slightly better over time for TIRPs discovered using EWD with two symbols per variable. In contrast, FCPM performed poorly over time for TIRPs discovered using EWD with four symbols per variable

on the graph represents the mean performance of the FCPM at a instances' revealed portion of time, averaging the results of the different TIRPs.

Figure 21 shows that CPMs provided more accurate predictions over time, in terms of AUROC, for TIRPs discovered using two symbols per variable or KB. In terms of AUPRC, two symbols and KB performed better for the CSP dataset. For the AHE, DBT, and EFIF datasets, four symbols per variable performed better than other numbers of symbols at making relatively early decisions, but as long as time went by, two symbols and KB closed the gap and performed better at making relatively late decisions. The poor performance of the three symbols per variable is related to the poor performance of the GRAD abstraction, which was tested only with three symbols. Looking at the overall mean results, CPMs provided more accurate predictions over time for TIRPs discovered using two symbols per variable or KB than three and four symbols per variable.

Lastly, we evaluated the performance of a different number of symbols (two, three, four, and a varied number of symbols for KB) only on TIRPs discovered using EWD and KB. Figure 22 presents the number of TIRPs per temporal abstraction and number of symbols and the mean AUROC and AUPRC result over the instances' revealed portions of time. Each point on the graph represents the mean performance of the FCPM, at a instances revealed portion of time, in providing the completions probabilities for the different TIRPs.

Figure 22 shows the CSP and DBT datasets that were abstracted with EWD with two symbols per variable resulting in more TIRPs that end with the target event than other temporal abstraction methods. In contrast, for the AHE dataset, there was much difference between the EWD or KB and the number of TIRPs. For the EFIF dataset, the different number of symbols per variable resulted in a similar number of TIRPs. EWD with two and four symbols per variable resulted in more accurate predictions over time. In addition, it can be shown that FCPM provided more accurate predictions over time, in terms of AUROC, for TIRPs discovered using KB for the CSP and AHE datasets. In contrast, for the DBT dataset, FCPM provided less accurate predictions with KB. In addition, in terms of AUPRC, FCPM performed poorly with KB for the AHE and DBT datasets. For the DBT dataset, EWD performed better than KB with all numbers of symbols. Overall, FCPM performed slightly better over time for TIRPs discovered using EWD with two symbols per variable. In contrast, FCPM performed poorly over time for TIRPs discovered using EWD with four symbols per variable.

In summary, the FCPM performed better than the baseline models in terms of AUROC and AUPRC. The AUPRC performance of the SCPM was poor at making relatively early decisions but improved as time went by. Moreover, FCPM provided less accurate predictions for TIRPs discovered using GRAD and more accurate predictions for TIRPs discovered using KB and EWD. While comparing the predictions over time provided by FCPM, TIRPs discovered that using two symbols per variable or KB led to better results. More specifically, FCPM performed slightly better over time for TIRPs discovered using EWD with two symbols per variable.



## References

1. Chang L, Wang T, Yang D, Luan H (2008) Seqstream: mining closed sequential patterns over stream sliding windows. In: 2008 Eighth IEEE international conference on data mining, pp 83–92. IEEE
2. Höppner F (2001) Learning temporal rules from state sequences. In: IJCAI workshop on learning from temporal and spatial data, vol 25. Citeseer
3. Papapetrou P, Kollios G, Sclaroff S, Gunopulos D (2009) Mining frequent arrangements of temporal intervals. Knowl Inf Syst 21(2):133

4. Mordvanyuk N, López B, Bifet A (2021) Verttirp: robust and efficient vertical frequent time interval-related pattern mining. Expert Syst Appl 168:114276
5. Harel O, Moskovitch R (2021) Complete closed time intervals-related patterns mining. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 4098–4105
6. Sacchi L, Larizza C, Combi C, Bellazzi R (2007) Data mining with temporal abstractions: learning rules from time series. Data Min Knowl Disc 15(2):217–247
7. Lu EH-C, Tseng VS, Philip SY (2010) Mining cluster-based temporal mobile sequential patterns in location-based service environments. IEEE Trans Knowl Data Eng 23(6):914–927
8. Li K, Fu Y (2014) Prediction of human activity by discovering temporal sequence patterns. IEEE Trans Pattern Anal Mach Intell 36(8):1644–1657
9. Moskovitch R, Shahar Y (2015) Classification-driven temporal discretization of multivariate time series. Data Min Knowl Disc 29(4):871–913
10. Patel D, Hsu W, Lee ML (2008) Mining relationships among interval-based events for classification. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data. ACM, pp 393–404
11. Batal I, Valizadegan H, Cooper GF, Hauskrecht M (2013) A temporal pattern mining approach for classifying electronic health record data. ACM Trans Intell Syst Technol 4(4):1–22
12. Itzhak N, Nagori A, Lior E, Schvetz M, Lodha R, Sethi T, Moskovitch R (2020) Acute hypertensive episodes prediction. In: International conference on artificial intelligence in medicine. Springer, pp 392–402
13. Novitski P, Cohen CM, Karasik A, Shalev V, Hodik G, Moskovitch R (2020) All-cause mortality prediction in t2d patients. In: International conference on artificial intelligence in medicine. Springer, pp 3–13
14. Teinemaa I, Dumas M, Leontjeva A, Maggi FM (2018) Temporal stability in predictive process monitoring. Data Min Knowl Disc 32(5):1306–1338
15. Teinemaa I, Dumas M, Rosa ML, Maggi FM (2019) Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans Knowl Discov Data 13(2):1–57
16. Di Francescomarino C, Ghidini C, Maggi FM, Milani F (2018) Predictive process monitoring methods: Which one suits me best? In: International conference on business process management. Springer, pp 462–479
17. Henry KE, Hager DN, Pronovost PJ, Saria S (2015) A targeted real-time early warning score (trewscore) for septic shock. Sci Transl Med 7(299):299–122299122
18. Schvetz M, Fuchs L, Novack V, Moskovitch R (2021) Outcomes prediction in longitudinal data: study designs evaluation, use case in icu acquired sepsis. J Biomed Inform 117:103734
19. Sheetrit E, Nissim N, Klimov D, Shahar Y (2019) Temporal probabilistic profiles for sepsis prediction in the icu. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, pp 2961–2969
20. Ryoo MS (2011) Human activity prediction: early recognition of ongoing activities from streaming videos. In: 2011 international conference on computer vision. IEEE, pp 1036–1043
21. Liu L, Wang S, Su G, Hu B, Peng Y, Xiong Q, Wen J (2017) A framework of mining semantic-based probabilistic event relations for complex activity recognition. Inf Sci 418:13–33
22. Zhu G, Cao J, Li C, Wu Z (2017) A recommendation engine for travel products based on topic sequential patterns. Multimed Tools Appl 76(16):17595–17612
23. da Silva Junior LLN, Kohwalter TC, Plastino A, Murta LGP (2021) Sequential coding patterns: how to use them effectively in code recommendation. Inf Softw Technol 140:106690
24. Huang C, Wang Y, Li X, Ren L, Zhao J, Hu Y, Zhang L, Fan G, Xu J, Gu X et al (2020) Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. The Lancet 395(10223):497–506
25. Itzhak N, Tal S, Cohen H, Daniel O, Kopylov R, Moskovitch R (2022) Classification of univariate time series via temporal abstraction and deep learning. In: 2022 IEEE international conference on big data (big data). IEEE, pp 1260–1265
26. Allen JF (1983) Maintaining knowledge about temporal intervals. ACM, New York
27. Moskovitch R, Shahar Y (2015) Fast time intervals mining using the transitivity of temporal relations. Knowl Inf Syst 42(1):21–48
28. Kujala R, Weckström C, Darst RK, Mladenović MN, Saramäki J (2018) A collection of public transport network data sets for 25 cities. Sci Data 5(1):1–14
29. Johnson AE, Pollard TJ, Shen L, Li-wei HL, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG (2016) Mimic-iii, a freely accessible critical care database. Sci Data 3:160035
30. Mirsky Y, Shabtai A, Rokach L, Shapira B, Elovici Y (2016) Sherlock vs moriarty: a smartphone dataset for cybersecurity research. In: Proceedings of the 2016 ACM workshop on artificial intelligence and security, pp 1–12
31. Höppner F (2002) Time series abstraction methods-a survey. In: GI Jahrestagung, pp 777–786

32. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery. ACM, pp 2–11
33. Bonomi L, Jiang X (2018) Pattern similarity in time interval sequences. In: 2018 IEEE international conference on healthcare informatics (ICHI). IEEE, pp 434–435
34. Ho N, Pedersen TB, Vu M, et al (2021) Efficient and distributed temporal pattern mining. In: 2021 IEEE international conference on big data (big data). IEEE, pp 335–343
35. Lee Z, Lindgren T, Papapetrou P (2020) Z-miner: an efficient method for mining frequent arrangements of event intervals. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining, pp 524–534
36. Zheng W, Hu J (2022) Multivariate time series prediction based on temporal change information learning method. IEEE Trans Neural Netw Learn Syst
37. Zheng W, Zhao P, Chen G, Zhou H, Tian Y (2022) A hybrid spiking neurons embedded lstm network for multivariate time series learning under concept-drift environment. IEEE Trans Knowl Data Eng
38. Ramírez-Gallego S, García S, Mouriño-Talín H, Martínez-Rego D, Bolón-Canedo V, Alonso-Betanzos A, Benítez JM, Herrera F (2016) Data discretization: taxonomy and big data challenge. Wiley Interdiscip Rev Data Min Knowl Discov 6(1):5–21
39. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Disc 15(2):107–144
40. Yi B-K, Faloutsos C (2000) Fast time sequence indexing for arbitrary lp norms. In: VLDB. Citeseer, vol 385, pp 99
41. Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2001) Dimensionality reduction for fast similarity search in large time series databases. Knowl Inf Syst 3(3):263–286
42. Freksa C (1992) Temporal reasoning based on semi-intervals. Artif Intell 54(1–2):199–227
43. Harris ZS (1954) Distributional structure. Word 10(2–3):146–162
44. Moskovitch R, Choi H, Hripcsak G, Tatonetti NP (2016) Prognosis of clinical outcomes with temporal patterns and experiences with one class feature selection. IEEE/ACM Trans Comput Biol Bioinf 14(3):555–563
45. Dvir O, Wolfson P, Lovat L, Moskovitch R (2020) Falls prediction in care homes using mobile app data collection. In: International conference on artificial intelligence in medicine. Springer, pp. 403–413
46. Moskovitch R, Walsh C, Wang F, Hripcsak G, Tatonetti N (2015) Outcomes prediction via time intervals related patterns. In: 2015 IEEE international conference on data mining. IEEE, pp 919–924
47. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J et al (2020) Scipy 1.0: fundamental algorithms for scientific computing in python. Nat Methods 17(3):261–272
48. Freedman D, Diaconis P (1981) On the histogram as a density estimator: L 2 theory. Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete 57(4):453–476
49. Kalbfleisch JD, Prentice RL (2011) The statistical analysis of failure time data, vol 360. Wiley, New York
50. Verduijn M, Sacchi L, Peek N, Bellazzi R, de Jonge E, de Mol BA (2007) Temporal abstraction for feature extraction: a comparative case study in prediction from intensive care monitoring data. Artif Intell Med 41(1):1–12
51. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
52. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133
53. Rumelhart DE, Hinton GE, Williams RJ (1985) Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science
54. Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on machine learning, pp 233–240
55. Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE 10(3):e0118432
56. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Icml

**Nevo Itzhak** is a Ph.D. student in the Department of Software and Information Systems Engineering at Ben-Gurion University of the Negev. He has been awarded the Jabotinsky scholarship in applied science for Ph.D. students by the Ministry of Science and Technology of Israel. Nevo holds a B.Sc. and an M.Sc., both awarded Summa cum Laude, in Software and Information Systems Engineering from Ben-Gurion University of the Negev. His research focuses on machine learning for temporal data, specifically continuous prediction and classification in heterogeneous multivariate temporal data. He has also served on program committees at several well-known artificial intelligence conferences.

**Szymon Jaroszewicz** is a professor at the Institute of Computer Science, Polish Academy of Sciences, where he heads the Statistical Analysis and Modeling Group. He is also with the Faculty of Mathematics and Information Science, Warsaw University of Technology. He received his Ph.D. from University of Massachusetts Boston in 2003, Doctor of Science degree from Institute of Computer Science, Polish Academy of Sciences in 2010, and the title of full professor in 2020. In 1998, he received a Fulbright Scholarship. His main research interest is uplift modeling which concerns building causal models based on treatment and control groups, but he is also active in other areas of data mining and statistical data analysis. He is an author of over 60 publications in those fields. For many years, he has been a member of program committees of several prominent data mining conferences and is a member of the editorial boards of Data Mining and Knowledge Discovery Journal and Fundamenta Informaticae.

**Robert Moskovitch**: Prof Moskovitch is heading the Complex Data Analytics Lab, as a faculty of the Department of Software and Information Systems Engineering at Ben Gurion University, Israel. Before his postdoc fellowship at the Department of Biomedical Informatics at Columbia University in NYC, he headed several R&D projects in Information Security at the Deutsche Telekom Innovation Laboratories. He is the vice president of the international society of Artificial Intelligence in Medicine (AIME), an Academic Editor at PLOS ONE, member of the editorial board of the Journal of Biomedical Informatics (JBI), and other. He was the co-chair of the international conference on Artificial Intelligence in Medicine (AIME) 2020. He serves on program committees of conferences, such as ACM KDD, IJCAI, AAAI, UAI, and AIME. He co-edited special issues at JASIST on Medical Information Retrieval, JBI on Temporal Data Analytics, and JAIR on AI and COVID-19, and AIMJ on AIME2020. He published more than hundred peer-reviewed papers in leading journals and conferences, such as IEEE ICDM, AAAI, Data Mining and Knowledge Discovery, Information Sciences, KAIS, JAMIA, and JBI, several of which had won best-paper awards.