2011 Eighteenth International Symposium on Temporal Representation and Reasoning

Temporal Functional Dependencies Based on Interval Relations

Carlo Combi
Department of Computer Science
University of Verona
Verona, Italy
Email: carlo.combi@univr.it

Pietro Sala
Department of Computer Science
University of Verona
Verona, Italy
Email: pietro.sala@univr.it

***Abstract*—In the last years the representation and management of temporal information has become crucial for several computer applications. In the temporal database literature, every fact stored into a database may be equipped with two temporal dimensions: the valid time, that describes the time when the fact is true in the modeled reality, and the transaction time, that describes the time when the fact is current in the database and it can be retrieved. Temporal functional dependencies (TFDs) add (transaction) valid time to classical functional dependencies (FDs) in order to express database integrity constraints over the flow of time. Currently, proposals dealing with TFDs adopt a point-based approach, where tuples hold at specific time points. Moreover, TFDs may involve the use of different granularities (i.e., partitions of the time domain), to express integrity constraints as "for each month, the salary of an employee depends only on his role". At the best of our knowledge, there are no proposals dealing with interval-based temporal functional dependencies (ITFDs for short) where the associated valid time is represented by an interval. In this paper, we propose a set of ITFDs based on the Allen's interval relations, we analyze their expressive power with respect to other TFDs proposed in the literature and we propose an algorithm for verifying ITFDs in a database system.**

I. Introduction

Temporal functional dependencies (TFDs) add a temporal dimension to classical functional dependencies (FDs) to deal with temporal data [1]–[5] (as a matter of fact, two temporal dimensions have been considered only in [6], where standard FDs are evaluated at every database snapshot). As an example, while FDs model constraints like "*employees with the same role get the same salary*", TFDs can represent constraints like "*for any given month, employees with the same role have the same salary, but their salary may change from one month to the next one*" [1], [4] or "*current salaries of employees uniquely depend on their current and previous roles*" [2]. To the best of our knowledge all the TFDs proposed in literature, including the ones cited above, rely on some point-based semantics or, like in the case of granularities, on some fixed point-based temporal grouping. On the other side, in a real world setting we are used to conceive time more like arbitrary intervals with a starting and an ending point in place of isolated time instants (e.g. *"my flight from Venice to Lübeck will take off at 10 o'clock and it will last 8 hours"*). Moreover, very often in our natural language descriptions two different time intervals are linked together by some relation (*"during the flight from Venice to Lübeck I red a nice book"*). Moving back from the real world to the enchanted land of temporal databases, it appears reasonable to analyze in which context the interval representation of time may result useful, if not mandatory in some case, for representing constraints over the possible evolutions of our data. In this paper, we propose a family of TFDs based on the Allen's interval relations [7] called Interval TFDs (ITFDs for short) and analyze their expressiveness by means of a simple example extracted from the clinical domain. Moreover we show that verifying whether a temporal database satisfies a given ITFD can be done in $\mathcal{O}(n \log n)$ (basically the cost of sorting the endpoints of the intervals involved). The paper is organized as follows. In Section II we give a short description of the literature on temporal functional dependencies. In Section III we introduce an example based on a real world scenario, the management of medical data in our case, that will be useful through the following sections, to give an idea of how TFDs and ITFDs work. In Section IV we introduce Interval Temporal Functional Dependencies and analyze their expressiveness. Section V describes a family of new algorithms for evaluating in an efficient way Interval Temporal Functional Dependencies over a temporal database. Finally, Section VI provides some concluding remarks and discusses further extensions of the current work.

II. Related work

In this section we first introduce two contributions, one from the temporal database area and another from the AI area, dealing with some semantic issues when associating data to intervals; then, we provide a short overview of the main formalisms for TFDs proposed in the literature.

A. Interval-based semantics

In [8] Terenziani and Snodgrass analyze the inadequacy of point-based semantics concerning models of natural language. They propose a dichotomy between two types of fact. Facts are partitioned in two classes, the telic ones (from the Greek "telos" meaning goal) and the atelic ones

1530-1311/11 $26.00 © 2011 IEEE
DOI 10.1109/TIME.2011.15

IEEE computer society

(the Greek 'a' as a prefix indicates negation). Telic events are characterized by the fact that they reach a *culmination* (e.g., "John won the lottery") while atelic facts do not have an intrinsic culmination (e.g., "John is building an house"). Moreover, in [8] Terenziani and Snodgrass propose an algebraic framework which deals with combinations of telic and atelic facts and they show how to add these concepts to a temporal query language (SQL/Temporal [9]).

In [10] Shoham proposes a first order logic for dealing with the truth of propositions among intervals. In particular, the author observes that the truth of a proposition over an interval is related to its truth over other intervals. The author classifies propositions depending on the relations that have to be considered in order to determine their truth. A proposition type x is *downward-hereditary* (written $\downarrow x$) if whenever it holds over an interval it holds over all of its sub-intervals, possibly excluding the two endpoints: for instance, *"John played less than fourty minutes"* is downward-hereditary. Symmetrically a proposition type x is *upward-hereditary* (written $\uparrow x$) if whenever it holds over all the sub-intervals of a given interval, possibly excluding the two endpoints, it also holds over the given interval itself: for instance, *"The airplane flies at* 35000 *feet"* is upward-hereditary. A proposition type x is *liquid* (written $\updownarrow x$) if it is both downward-hereditary and upward-hereditary (e.g. *the room is empty*). A proposition type x is *concatenable* if whenever it holds over two consecutive intervals it holds also over their union, for instance the proposition *"John travelled an even number of miles"*. A proposition type x is *gestalt* if whenever it never holds over two intervals one of which properly contains the other, for instance the proposition *"Exactly six minutes passed"* is gestalt. A proposition type x is *solid* if whenever it never holds over two properly overlapping intervals; for instance, the proposition *"The plane executed the LANDING procedure (from start to finish)"* is solid. As we will see in the following, some interesting properties that can be expressed using Shoham's proposition types cannot be captured by a point-based formalism.

B. A quick tour in Temporal Functional Dependencies

In this section we illustrate the most important (point-based) temporal functional dependencies proposed in the literature. In [6], [11] Jensen et al. propose a bitemporal data model that allows one to associate both valid and transaction times with data. The basic atomic entity of Jensen's TFDs is the so called *bitemporal chronon* which is an ordered pair consisting of a valid time and of a transaction time. The schema of atemporal attributes is extended with a bitemporal chronon. Jensen et al.'s TFDs make it possible to express conditions that must be satisfied at any (bitemporal) time point taken in isolation. Bettini, Jajodia, and Wang's notion of TFD takes advantage of time granularity [12], [1]. Examples of granularities are Day, $Month$, and $WorkingDay$; a granule is an element of a given granularity (e.g. *"November 2010"* is a granule of the granularity $Month$). A database schema R is extended to a *temporal module schema* which is a triple (R, G, ϕ), where G is a granularity and ϕ is a function, called time windowing function, that associates every tuple with the granules in G where the tuple is valid. Bettini, Jajodia, and Wang's TFDs allow one to specify conditions on tuples associated with granules of a given granularity and grouped according to a coarser granularity. A general formalism for TFDs on complex (temporal) objects has been proposed by Wijsen in [4]. It is based on a data model that extends the relational model with the notion of object identity, which is preserved through updates, and with the ability of dealing with complex objects, that is, objects that may have other objects as components. The time domain is assumed to be (isomorphic to) $\mathbb{N}$. A time relation is a subset of $\mathbb{N} \otimes \mathbb{N}$. A time granularity can be defined as a special case of time relation, called *chronology*. For example, the granularity $Month$ can be defined as the smallest set of pairs (i, j), where i and j belong to the same month and $i \leq j$. Wijsen TFDs are written as $c : X \rightarrow_{\alpha} Y$ and can be intuitively explained as follows. Let t_1 and t_2 be two objects of class c at time points i and j, respectively, where (i, j) belongs to the time relation α. If t_1 and t_2 agree on X, then they must agree on Y as well. It is not difficult to show that the class of Wijsen's TFDs subsumes the class of Bettini et al.'s TFDs. More precisely, Bettini et al.'s TFDs are exactly all and only the TFDs on chronologies (the class TFD-C in Wijsen's terminology). In an earlier work, Wijsen introduces a special notation for some relevant subclasses of TFDs [3]. In particular, he abbreviates $X \rightarrow_{next} Y$ as $X\mathbf{N}Y$ and for $X \rightarrow_{Forever} Y$ as $X\mathbf{G}Y$. Wijsen's TFDs allow one to specify conditions on tuples grouped according to any given time relation. In [2] Vianu proposes a simple extension to the relational model in order to describe the evolution of a database over time. According to it, a temporal database is viewed as a sequence of instances (states) over time. A change in the state of the database is produced by the execution of an update, an insertion, or a deletion. A *database sequence* is a sequence of consecutive instances of the database, together with *"update mappings"* from one instance (the "old" one) to the next instance (the "new" one). Tuple are viewed as representations of domain objects. Properties of the evolution of objects over time are expressed by "dynamic" dependencies (DFDs), which are defined by means of "action relations" associated with updates. Intuitively, an action relation is generated by concatenating each tuple in the "old" instance with its updated version.

III. A MOTIVATING EXAMPLE

In the following we propose a little example of expressivity of the TFDs described in section II and we show an interesting constraint that cannot be expressed through point-based TFDs.

#	Therapy	PatId	Phys	Drug	Qty	B	E
1	antiviral	1	Dorian	acyclovir	300	1	16
2	analgesics	1	Cox	paracetamol	200	2	10
3	cardiovascular	1	Turk	atenolol	100	3	8
4	antipyretics	1	Cox	paracetamol	100	9	11
5	sedative	1	Turk	diazepam	10	13	15
6	anxiolytic	1	Cox	diazepam	10	17	19
7	antiviral	2	Kelso	acyclovir	200	1	10
8	cardiovascular	2	Quinlan	atenolol	100	4	7
9	analgesics	2	Reid	paracetamol	150	5	9
10	antiviral	2	Reid	acyclovir	300	9	14

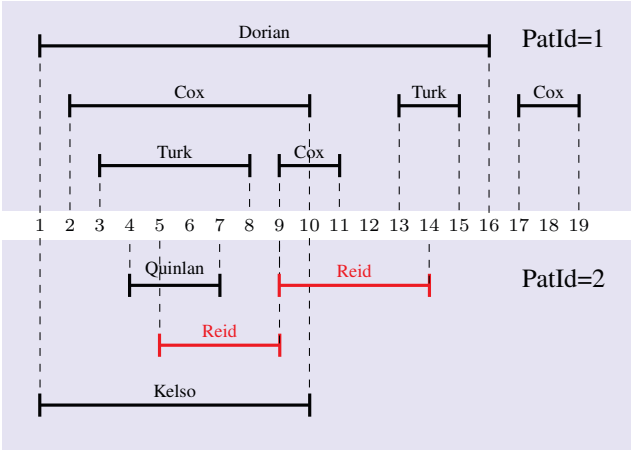

Figure 1. An instance s of the relation $Patient$, storing data about therapies.

Most health care institutions collect a large quantity of clinical information about patients and physician's actions, such as therapies and surgeries, as well as about health care processes, such as, admissions, discharges, and exam requests. All these pieces of information are temporal in nature and the associated temporal dimension needs to be carefully considered, in order to be able to properly repre- sent clinical data and to reason on them. In this section, we briefly introduce a real-world example taken from clinical medicine, namely that of patient therapies. Suppose to have patients which undergo several and different therapies: each therapy can be supervised by a physician, and consists of the administration of some drug to the patient. Information about patients and therapies are stored in a relational schema $Patient = (Therapy, PatId, Drug, Qty, Phys, B, E)$, where $Therapy$ identifies a medical treatment, $PatId$ represents fa patient ID, $Drug$ and Qty the drug prescribed and its quantity, respectively, and $Phys$ the physician who made the prescription (and is responsible of the therapy). Finally, attributes B and E represent the starting and ending time points of the tuple validity interval, respectively: they represent the bounds of the interval specified by the physician for each therapy. An instance of relation $Patient$ is provided in Figure 1.

As an example, on this relation the requirement *"at any time, the quantity of the prescribed drug depends on the type of drug and on the current therapy"* can be expressed by Jensen's TFDs, while the requirement "*every month, the physician who prescribes a given drug depends on the therapy* " may be captured by Bettini, Jajodia, and Wang's TFDs. Moreover, it is possible to express by Wijsen's TFDs the requirement *" for every patient, the quantity of a drug cannot change within 16 days"*: it means that we have to wait for 16 days without prescribing a drug to a patient, if we want to change the quantity of that drug for the given patient. It is worth to notice that this kind of constraints cannot be expressed by the Bettini, Jajodia, and Wang's TFDs, since every granule cannot overlap another one. Finally, the requirement:*"the new quantity for a drug depends only on the old quantity"* may be captured by Vianu's DFDs.

Let us suppose that our database has to respect the following constraint.

Example 1. The policy of the hospital is the following:

1) every patient may receive several therapies at the same time from different physicians;
2) overlapping therapies for the same patient must be prescribed by the same physician (in other words, if a patient during a therapy needs another therapy which lasts after the end of the current therapy, then this therapy must be prescribed by the current physician).

It is easy to see that in order to ensure this condition both the starting points and the ending points of every pair of tuples come into play. Thus, the point-based TFDs proposed in Section II-B cannot be used to specify the above requirement related to the hospital policy.

IV. Interval-based Functional Dependencies

In this section we propose a new type of temporal functional dependency based on Allen's interval relations.

A. Interval relations

Given a linear order $\mathbb{O} = \langle O, < \rangle$, an interval I over $\mathbb{O}$ is a pair $I = [b, e]$ where $b, e \in O$ and $b \leq e$. Given an interval $I = [b, e]$ over $\mathbb{O}$ we identify with $points(I)$ the set of points in O between b and e: $points(I) = \{p \mid p \in O \text{ and } b \leq p \leq e\}$. While the possible distinct relations between two points considering only the linear order are reduced to three (equality, successor, and predecessor), considering the order among the two endpoints of two intervals leads us to have

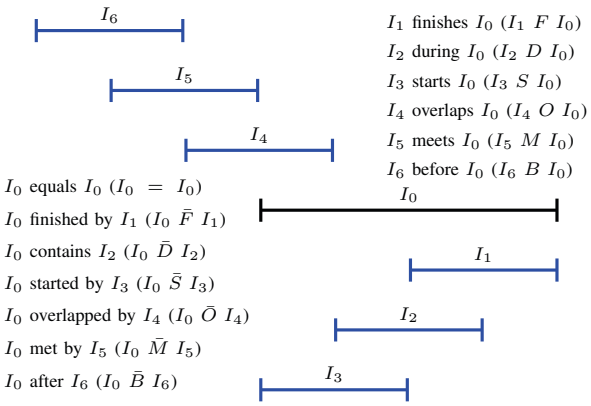

Figure 2. The thirteen Allen's relations between intervals.

thirteen possible relations. These relations are depicted in Figure 2 according to the notation proposed by Allen in [7]. It is worth to note that every relation has its dual which is obtained by switching the position of the two intervals. Consider, for example, two intervals $I_1 = [b_1, e_1]$ and $I_2 = [b_2, e_2]$: we have that $I_1\ D\ I_2$ (I_1 *during* I_2), if and only if $b_2 < b_1 < e_1 < e_2$; by reverting the arguments, we have that $I_2\ \overline{D}\ I_1$ (I_2 *contains* I_2), if and only if $b_2 < b_1 < e_1 < e_2$, which is equivalent to $I_1\ D\ I_2$. More precisely, given two intervals $I_1 = [b_1, e_1]$ and $I_2 = [b_2, e_2]$ we say that:

(1) $I_1\ M\ I_2$ iff $e_1 = b_2$;
(2) $I_1\ S\ I_2$ iff $b_1 = b_2$ and $e_1 < e_2$;
(3) $I_1\ F\ I_2$ iff $b_1 > b_2$ and $e_1 = e_2$;
(4) $I_1\ O\ I_2$ iff $b_1 < b_2$ and $b' < e_1 < e_2$;
(5) $I_1\ D\ I_2$ iff $b_2 < b_1$ and $e_1 < e_2$;
(6) $I_1\ B\ I_2$ iff $e_1 < b_2$.

In the following we will consider the (sub) set $\mathcal{A} = \{M, S, F, O, D, B\}$ of Allen's interval relations (without considering the dual ones and the equality relation).

B. The interval-based temporal relational data model

In discussing our new functional dependencies based on intervals within a relational framework, we will use a simple temporal (relational) data model based on the concept of temporal relation. A temporal relation $\mathbf{r}$ is a relation on a temporal relation schema $\mathcal{R}$ defined on the attributes $U \cup \{B, E\}$, where U represents a set of atemporal attributes and B, E are the temporal attributes describing the valid interval of a tuple. We assume that the domain of both the attributes B and E is a totally ordered set $\mathbb{O}$. Clearly a tuple $t \in \mathbf{r}$ satisfies $t[B] \leq t[E]$. We use the notation $\mathbf{att}(\mathcal{R})$ to denote the set of attributes (both atemporal and temporal) of the relation schema $\mathcal{R}$ and $\mathbf{att}^{-}(\mathcal{R})$ to denote the set of its atemporal attributes. We recall that equality (inequality) atomic formulas are expressions either of the form $t[A] = w[B]$ ($t[A] \neq w[B]$) or of the form $t[A] = a$ ($t[A] \neq a$), being A, B attribute names, a a constant value and w, t tuples of relation $\mathbf{r}$. Moreover, assuming that the underlying domain for attributes A and B has a total order, comparison atomic formulas are either of the form $t[A]\theta w[B]$ or of the form $t[A]\theta a$, with $\theta \in \{<, \leq, >, \geq\}$. When we consider a temporal relation, it is important to denote the set of tuples valid "at" a specific time interval (i.e., the values of attributes B and E of the tuples match exactly with the endpoints of the interval, respectively). Given a time interval $I = [b, e]$ on $\mathbb{O}$, the snapshot at interval I of the temporal relation $\mathbf{r}$, denoted with $\mathbf{r}_I$, is the relation containing all and only the tuples of $\mathbf{r}$ having valid interval $I = [b, e]$. More formally, the snapshot at interval I of $\mathbf{r}$ is obtained as $\mathbf{r}_I = \{t \mid \mathbf{r}(t) \wedge t[B] = b \wedge t[E] = e\}$.

C. ITFDs

Let us now consider the basic definition of *Interval-based Temporal Functional Dependency* (ITFD). In the following, we will consider only interval relations in the set $\mathcal{A}$: indeed, in this case it is not meaningful to distinguish between a relation and its dual, as it will be clear from the following definition of interval-based temporal functional dependency.

Definition IV.1. Let X and Y be sets of atemporal attributes of a temporal relation schema $\mathcal{R} = R(U, B, E)$ and $\sim_I \in \mathcal{A}$ an Allen's Interval relation. A database instance $\mathbf{r}$ of $\mathcal{R}$ satisfies an ITFD $X \rightarrow_{\sim_I} Y$ iff

$$\forall b, b', e, e' \in O \text{ with } b \leq e\ ,\ b' \leq e' \text{ and } [b, e] \sim_I [b', e']$$

$$\text{and } \forall s_1, s_2 \in \{t \mid r^B(t) \wedge ((t[B] = b \wedge t[E] = e) \vee (t[B] = b' \wedge t[E] = e'))\}$$

we have $s_1[X] = s_2[X] \Rightarrow s_1[Y] = s_2[Y]$.

Basically ITFDs group tuples whose B and E attribute values satisfy the interval relation $\sim_I$. In the above definition all the possible tuples having as valid interval either $[b, e]$ or $[b', e']$, where $[b, e] \sim_I [b', e']$ are considered together. If there exist two tuples where the B and E attribute values match exactly the points b, e, b', and e', respectively, and both tuples agree on the values of atemporal attributes X, then the ITFD imposes that both the tuples must agree on the values of atemporal attributes Y. Consider the medical relation schema $Patient$ proposed in Section III and suppose that we want to express the constraint *"prescriptions of a given therapy starting the same day must have the same physician"*: such a requirement is expressed by the ITFD $Therapy \rightarrow_{\mathbf{S}} Phys$. For a more significant example of the expressivity of our ITFDs we can recall the scenario depicted in Section III which can be rephrased as *"overlapping drug administrations for a given patient must have the same physician"* (as specified in Example 1). This constraint can be expressed by the ITFD $PatId \rightarrow_O Phys$. A time-oriented graphical account of tuples of relation *Patient* is provided in lower part of Figure 1.

As we may notice, the relation satisfies the ITFD $PatId \rightarrow_O Phys$ only for tuples related to the patient with $PatId = 1$: Dr. Cox added a therapy *antipyretics*, but the related valid interval is contained in the interval of therapy $antiviral$, prescribed by Dr. Dorian. Tuples related to therapies of patient with $PatId = 2$, instead, do not satisfy ITFD $PatId \rightarrow_O Phys$, as both the intervals of therapies prescribed by Dr. Reid overlap a therapy prescribed by another physician. It is easy to show that this kind of properties cannot be expressed through point-based TFDs. Basically this lack of expressiveness depends on the fact that point-based TFDs refer only to database snapshots which are either evaluated in isolation towards TFDs or grouped together according to some granularity or joined to the next snapshot to consider some kind of tuple evolution. In our example, intervals of prescriptions are considered in a holistic way and the considered temporal dependency is not checked against all the database snapshots, as it has to consider the specified interval relation. In our example *Dorian* starts a therapy on the patient with $PatId = 1$ and Cox starts therapy *analgesics* for the same patient during this therapy. After that, *Cox* adds another therapy (*antipyretics*). Suppose that this last therapy of *Cox* would last until time 14: then, either this tuple or the tuple related to therapy by *Turk* having valid time $[13, 14]$ violate the ITFD. Suppose that the database must satisfy the ITFD $PatId \rightarrow_O Phys$, and that tuples are inserted according to the start of their valid time: in this case, the insertion of tuple #5 would be blocked. Another constraint that can be expressed using ITFDs is *"intersecting therapies made by the same physician must have the same prescription"*. It is easy to see that the ITFD $Therapy, Phys \rightarrow_{\{S,F,O,D\}} Drug$ capture this property.

V. Evaluating ITFDs

In this section we describe a set of algorithms, one for each Allen's interval relation, we designed for evaluating ITFDs. First, we recall the problem of evaluating ITFDs. Let $X \rightarrow_{\sim_I} Y$, where $X \cup Y \subseteq U$ and $R \in \{M, S, F, O, D, B\}$ be an ITFDs, moreover it is easy to show that if an ITFD $X \rightarrow_{\sim_I} Y$ holds over some temporal relation **r** of some schema $\mathcal{R}$ then the ITFD $X \rightarrow_{\sim_{\overline{I}}} Y$ holds over **r** as well. Hence we can ignore the ITFDs based on the inverse relations and the $X \rightarrow_= Y$ relation which is trivial to verify, it suffices to group tuples with the same values on the attributes B and E and verify that the FD $X \rightarrow Y$ holds on every group. Given a set of tuples $\mathcal{T}$ where for $\mathbf{att}(\mathcal{T}) = X \cup Y \cup \{B, E\}$, we say that $\mathcal{T}$ satisfies f if and only if for each pair of tuples $t, t' \in \mathcal{T}$, if we have $t[X] = t'[X]$ and $[t[B], t[E]]\ R\ [t'[B], t'[E]]$, then $t[Y] = t'[Y]$. It is worth to notice that given an ITFD $X \rightarrow_{\sim_i} Y$ and a set of tuples $\mathcal{T}$, we can create a partition $\mathcal{T}_1, ..., \mathcal{T}_n$ of $\mathcal{T}$ where for each $1 \leq i, j \leq n$ and for each $t \in \mathcal{T}_i$ and $t' \in \mathcal{T}_j$, we have $t[X] = t'[X]$ if and only if $i = j$. This partition can be easily obtained: it is enough to give an arbitrary linear order on each attribute $U \in X$ and an arbitrary linear order on attributes of X, and then sort the tuples using this new lexicographic order using some well-known $O(n \log n)$ sorting algorithm. After this preliminary step we have to verify f over every $\mathcal{T}_i$; the applied algorithm depends on the relation R. In the following we propose an algorithm for each relation $R \in \{M, S, F, O, D, B\}$. We propose an ad hoc implementation because checking

```
Verify-B(β)
  //β is lexicographically ordered on (B, E)
  verified ← |β|; i ← 1;
  while i < verified
  do { j ← search_begin(β, β_i[E]);
       if j < |β| + 1
       then { if β_i[Y] ≠ β_j[Y]
              then return false ;
              for k ← j + 1 to verified
              do { if β_i[Y] ≠ β_k[Y]
                   then return false ;
              if verified > j
              then verified ← j;
       i ← i + 1
  return true ;
```

Figure 3. The algorithm verifying B-based ITFDs.

```
Verify-M(β, ε)
  //β is lexicographically ordered on (B, E)
  //ε is lexicographically ordered on (E, B)
  α ← retrieve_endpoints(β);
  for i ← 1 to |α|
  do { j ← search_begin(β, α_i);
       k ← search_end(ε, α_i);
       if j < |β| + 1 ∧ k < |ε| + 1
       then { j' ← j + 1 :
              while β_j'[B] = β_j[B]
              do
              { if β_j'[Y] ≠ β_j[Y]
                then return false
                j' ← j' + 1;
              k' ← k :
              while ε_k'[E] = ε_k[E]
              do { if ε_k'[Y] ≠ β_j[Y]
                   then return false
                   k' ← k' + 1;
  return true
```

Figure 4. The algorithm verifying M-based ITFDs.

the proposed ITFDs represents a slightly different (and more simple) problem with respect to that of constraint propagation over interval networks faced by Allen in ([7]): as we will see, it is in a lower complexity class. In the following when we propose the algorithm for verifying if some ITFDs $X \rightarrow_{\sim_I} Y$ holds over some temporal relation **r** of some schema $\mathcal{R}$ we assume that the input tuples for such algorithm agree on the values for the attributes in X. If it is not the case we can partition the tuples in our relation **r** accordingly to the values of X and then we apply the algorithm on every partition. It is easy to see that the these partitions can be created starting from **r** using any well-known sorting algorithm with $\mathcal{O}(n \log n)$ complexity. This algorithm is used to sort the tuples of **r** lexicographically on X, after that we can build the partitions by simply split the ordered vector of tuples into blocks (this can be done in linear time). Summing up the complexity of this preliminary step does not exceed $\mathcal{O}(n \log n)$ which is the complexity of the algorithms proposed. In the following we assume, that n is the number of tuples present in the input for every algorithm proposed.

A. Evaluating $\{S, F, B, M\}$-based ITFDs

We start with the relation S and F which are the most trivial ones to be evaluated; nevertheless, the proposed algorithm contains the basic ingredients useful to describe the following algorithms. For verifying $X \rightarrow_S Y$ over a set of tuples $\mathcal{T}$ (**att**$(\mathcal{T}) \supseteq X \cup Y \cup \{B, E\}$) sharing the same values for the attribute set X it is sufficient to sort the tuples lexicographically on (B, E) using some well-known $O(n \log n)$ sorting algorithm. Let β the vector containing the tuples of $\mathcal{T}$ sorted lexicographically on (B, E) we refer to its i-th element as β_i for every $1 \leq i \leq |\beta|$. We return that $X \rightarrow_B Y$ is verified over $\mathcal{T}$ if and only if for every $1 \leq i < |\beta|$ the condition $\beta_i[B] = \beta_{i+1}[B]$ implies $\beta_i[Y] = \beta_{i+1}[Y]$ (this operation can be performed in linear time). In a symmetric way we can verify $X \rightarrow_F Y$ by taking the lexicographic order of the tuples of $\mathcal{T}$ on (E, B) stored in a vector ϵ. We return that $X \rightarrow_S Y$ is verified over $\mathcal{T}$ if and only if for every $1 \leq i < |\epsilon|$ the condition $\epsilon_i[E] = \epsilon_{i+1}[E]$ implies $\epsilon_i[Y] = \epsilon_{i+1}[Y]$. Suppose that we have to verify $X \rightarrow_B Y$ then we calculate the vector β and we launch the procedure in Figure 3.a. The procedure considers the intervals from the one belonging to the first position of β and verifies if for every interval which begins after the end of the current interval shares the same values for the attributes in Y. The function $search\_begin$ takes a lexicographically ordered vector β of intervals and a point p and return the first position of β which contains an interval that begins after p, if such a position does not exist then $search\_begin$ returns $|\beta| + 1$. It is easy to see that this is a simple search in a ordered vector then the complexity is $\mathcal{O}(\log(n))$. In order to avoid the repeated check of the same interval which will lead to a quadratic

```
Verify-O(β, ϵ)
  //β is lexicographically ordered on (B, E)
  //ϵ is lexicographically ordered on (E, B)
  main
  |α| ← 1; α_1.low ← 1; α_1.up ← 1;
  α_1.status ← free;
  α_1.max ← β_1[E]; α_1.begin ← β_1[B];
  i ← 2; j ← 1; k ← 1;
  while i ≤ |β|
  do
    if β_i[B] = α_k.begin
    then
      α_k.up = i;
      α_k.max = β_i[E];
      i ← i + 1;
    else
      if β_i[B] ≥ ϵ_j[E]
      then
        if ¬up_clusters(β, ϵ, α, j)
          then return false
        k ← |α|;
        j ← j + 1;
      else
        |α| ← |α| + 1;
        k ← k + 1; α_k.low ← i;
        α_k.up ← i;
        α_k.status ← free;
        α_k.max ← β_i[E];
        α_k.begin ← β_i[B];
        i ← i + 1;
  return true ;
```

Figure 5. The algorithm for verifying the interval relation O.

complexity, we introduce a variable $verified$, this variable keeps track of the interval which has already been checked. Since every position can be checked at most 2 times we have that the complexity of the procedure is $\mathcal{O}(n \log(n))$. Consider now the M (meets) operator and suppose to have calculated both the vectors β and ϵ, for every point p if there exists two tuples $t, t' \in \mathcal{T}$ with $t[B] = t'[E] = p$ then all the tuples $\bar{t} \in \mathcal{T}$ which satisfy $\bar{t}[B] = p$ or $\bar{t}[E] = p$ must share the same value for the attributes in Y. More precisely the procedure for checking an ITFD of the form $X \rightarrow_M Y$ is represented in Figure 3.b. This procedure first calculates a vector α representing the set of all the endpoints of all intervals in β by means of the function $retrieve\_endpoints$. This function is designed in a way that the vector α does not contain repetitions, and thus the complexity of $retrieve\_endpoints$ is $\mathcal{O}(n \log(n))$. The function $search\_end$ is the analogous of the function $search\_begin$ considering the vector ϵ. The procedure for every endpoint α_i in α considers two blocks of consecutive tuples one consisting of the tuples t in β with $t[B] = \alpha_i$ and the other consisting of the tuples t' in ϵ with $t'[E] = \alpha_i$. If both these two blocks are not empty, the procedure verifies

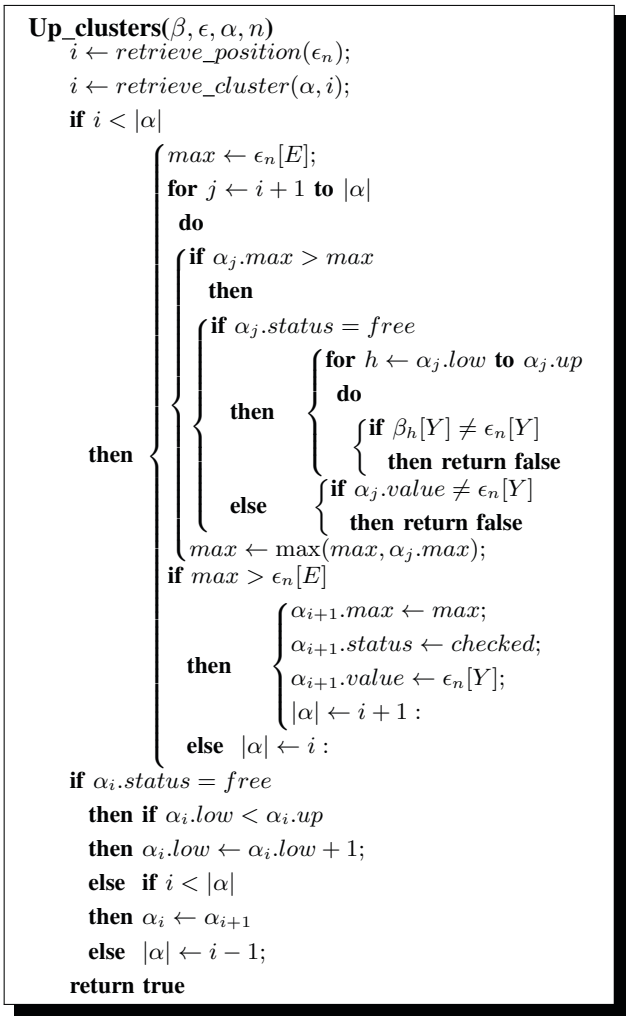

Figure 6. The auxiliary function $up\_cluster$ for the algorithm of Figure 5.

if all the tuples of these two blocks share the same values for attributes in Y. For evaluating the complexity it suffices to consider that every tuple is checked by the procedure at most two times (one for its right endpoint and one for its left one) and the number of evaluations of the functions $search\_begin$ and $search\_end$ is bounded by the number of endpoints, then the total complexity is $\mathcal{O}(n \log(n))$.

B. Evaluating $\{O, D\}$-based ITFDs

Now we consider the more complicated cases of verifying ITFDs $X \rightarrow_O Y$ and $X \rightarrow_D Y$. Informally our idea consists of exploring the space of intervals following the order of the endpoints associated to the tuples in $\mathcal{T}$. We use a queue to keep track of the active intervals at every step. At every step we update the queue by removing the tuples which end at the current point and, then, by adding the tuples which begin in the current point in between this two operations we perform consistency check between the tuples removed and the tuples which remains in the structure. It is worth to notice that by construction if a tuple t is inserted after a tuple t' in the queue then we have $t'[B] \leq t[B]$ then the queue is lexicographically sorted on the attributes B and E. Let t be a tuple we must remove at the current step: for every tuple t' with $t[B] < t'[B]$ which lies above t in the queue and it is not removed in the current step (which means $t[E] < t'[E]$), we have to verify that $t'[Y] = t[X]$. After the removal of all the tuples which end in the current point we can add the tuples which begin on the current point to the top of the queue. We give the intuition behind the proposed algorithm by means of the example depicted in Figure 7. Suppose that you want to verify the ITFD $X \rightarrow_O Y$ on the set of tuples $\mathcal{T} = \{A_0, A_1, B_0, B_1, C_0, C_1, C_2, D_0\}$ (assume that for all $T \in \mathcal{T}$ we have $t[X] = t[Y]$). At step 1 both tuples A_1 and A_0 are inserted in the queue and at step 2 the tuples B_0 and B_1 are inserted on the top of the queue. At step 3 first we remove B_0, since all the intervals above B_0 in the queue consist of the singleton B_1 which shares the same starting point with B_0 and then it is not yet checked for consistency. Step 3 terminates with the addition of the tuples C_0, C_1 and C_2 on the top of the queue. At step 4 tuples B_1 and C_0 are removed since B_1 and C_0 are removed in the same step they can differ on the values for the attributes Y. Since C_0 shared the same beginning point with all the tuples above it on the queue, then it can differ from C_1 and C_2 on the values for the attributes Y. When we remove B_1, tuples C_1, C_2 are still present in the queue and $B_1[B] < C_1[B] = C_2[B]$ then we have to check $B_1[Y] = C_1[Y] = C_2[Y]$. At step 5 we insert D_0 at the top of the queue. At step 6 the tuple C_1 is removed, as we said before C_2 is not checked for consistency because it shares the same beginning point with C_2 then only D_0 is checked for consistency and we verify $C_1[Y] = D_0[Y]$. At step 7 the tuple D_0 is removed without check anything since it is on the top of the queue. At step 8 the tuple A_0 is removed and since the tuple C_2 is not removed then it is checked for consistency with A_0 ($A_0[Y] = C_2[Y]$). Summing up the results at the end of the procedure we have that the ITFD $X \rightarrow_O Y$ is respected by $\mathcal{T}$ if and only if $B_1[Y] = C_1[Y] = C_2[Y] = D_0[Y] = A_0[Y]$ and it is worth to notice that that all these intervals do not pairwise overlap (consider B_1 and D_0 for instance). The procedure for verifying if the ITFDs $X \rightarrow_O Y$ over a set of tuples which agree on the attributes X is given in Figure 5. In order to give an efficient procedure for verifying $X \rightarrow_O Y$ we partition the vector β into *clusters*. A *cluster* is the maximal sequence of lexicographically ordered intervals which share the same beginning endpoint, it is straightforward to see, since β is ordered, that a cluster is a sequence of consecutive positions in β and thus can be represented as an interval on the positions of β. The correct management of clusters is guaranteed by the function $Up\_Clusters$ of Figure 6. The procedure for checking ITFDs like $X \rightarrow_D Y$ operates in a very symmetric way with respect to the procedure for

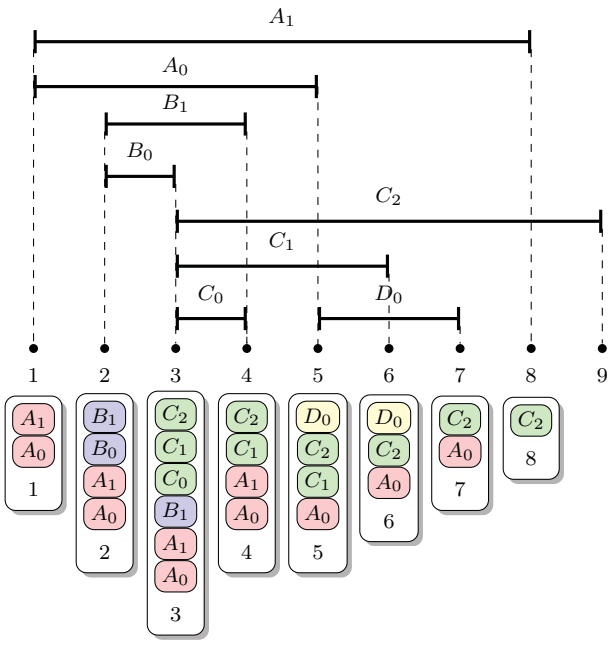

Figure 7. An example of the execution of the algorithm for the O and the D ITFDs.

$X \rightarrow_O Y$: it suffices to rewrite the algorithm in Figure 5 in order to look downward in the queue instead of upward. We can conclude this section with the following theorem.

Theorem V.1. *For every set of atemporal attributes X and Y of a temporal relation schema $\mathcal{R} = R(U, B, E)$ and for every Allen's Interval relation $\sim_I \in \mathcal{A}$ verifying if an instance $\boldsymbol{r}$ of $\mathcal{R}$ satisfies an ITFD $X \rightarrow_{\sim_I} Y$ takes at most $\mathcal{O}(|\boldsymbol{r}| \log(|\boldsymbol{r}|))$ steps.*

This result follows from the proofs of correctness and completeness for the algorithms proposed in subsections V-A and V-B which can be found in [13].

VI. Conclusions and future work

In this paper we proposed a set of interval-based temporal functional dependencies and discussed its expressiveness by means of an example taken from the clinical domain. Moreover, for each interval relation $\sim_I$ we proposed an algorithm to verify, given an instance **r** of a temporal database, whether an ITFD based on $\sim_I$ holds over **r**. We plan to implement the algorithm proposed and evaluate it over different real-worlds scenarios. A further step towards interval-based temporal functional dependencies will be devoted to extend the proposed ITFDs to deal with multiple temporal granularities and with interval-based tuple evolutions, similarly to the Vianu's (point-based) approach. As for the clinical domain, we plan to adapt and extend the proposed techniques to the data mining issue: indeed, in the clinical domain there is the need of mining temporal association rules among temporal data, often charachterized by intervals of validity. ITDBs may be considered as a case of interval-based temporal association rule.

References

[1] C. Bettini, S. G. Jajodia, and S. X. Wang, *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2000.

[2] V. Vianu, "Dynamic functional dependencies and database aging," *J. ACM*, vol. 34, no. 1, pp. 28–59, 1987.

[3] J. Wijsen, "Design of temporal relational databases based on dynamic and temporal functional dependencies," in *Temporal Databases*, 1995, pp. 61–76.

[4] ——, "Temporal fds on complex objects," *ACM Trans. Database Syst.*, vol. 24, no. 1, pp. 127–176, 1999.

[5] ——, "Temporal dependencies," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer US, 2009, pp. 2960–2966.

[6] C. S. Jensen, R. T. Snodgrass, and M. D. Soo, "Extending existing dependency theory to temporal databases," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 4, pp. 563–582, 1996.

[7] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[8] P. Terenziani and R. T. Snodgrass, "Reconciling point-based and interval-based semantics in temporal relational databases: A treatment of the telic/atelic distinction," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 5, pp. 540–551, 2004.

[9] R. T. Snodgrass, Ed., *The TSQL2 Temporal Query Language*. Kluwer, 1995.

[10] Y. Shoham, "Temporal logics in ai: Semantical and ontological considerations," *Artif. Intell.*, vol. 33, no. 1, pp. 89–104, 1987.

[11] C. S. Jensen and R. T. Snodgrass, "Temporally enhanced database design," in *Advances in Object-Oriented Data Modeling*, 2000, pp. 163–193.

[12] X. S. Wang, C. Bettini, A. Brodsky, and S. Jajodia, "Logical design for temporal databases with multiple granularities," *ACM Trans. Database Syst.*, vol. 22, no. 2, pp. 115–170, 1997.

[13] C. Combi and P. Sala, "Temporal functional dependencies based on interval relations," Department of Computer Science, University of Verona, Verona, Italy, Tech. Rep. RR 82/2011, 2011.