Dipartimento di Informatica
Università degli Studi di Verona

Rapporto di ricerca **81/2011**
Research report
March 2011

A uniform framework for temporal functional dependencies with multiple granularities

Carlo Combi

Dipartimento di Informatica, Università degli Studi di Verona
strada le Grazie 15, 37134 Verona Italy
carlo.combi@univr.it

Angelo Montanari

Dipartimento di Matematica e Informatica, Università degli Studi di Udine
via delle Scienze 206, 33100 udine Italy
angelo.montanari@uniud.it

Pietro Sala

Dipartimento di Informatica, Università degli Studi di Verona
strada le Grazie 15, 37134 Verona Italy
pietro.sala@univr.it

Questo rapporto è disponibile su Web all'indirizzo:
This report is available on the web at the address:
http://www.di.univr.it/report

Abstract

Temporal functional dependencies (TFDs) add a temporal component to classical functional dependencies to deal with temporal data. As an example, while functional dependencies 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*" or "*current salaries of employees uniquely depend on their current and previous roles*". In this paper, we propose a general framework for specifying TFDs, possibly involving different time granularities, and for checking whether or not a given database instance satisfies them. The proposed framework subsumes existing formalisms for TFDs and it allows one to encode TFDs which are not captured by them.

Keywords: Temporal Database, Functional Dependencies, Granularity, Biomedical Data.

1 Introduction

Temporal functional dependencies (TFDs) add a temporal dimension to classical functional dependencies (FDs) to deal with temporal data [2, 10, 11, 12, 13] (as a matter of fact, two temporal dimensions have been considered only in [5], 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*" [2, 12] or "*current salaries of employees uniquely depend on their current and previous roles*" [10]. Since temporal constraints may refer to different time units, e.g., university courses are organized on semesters, the scheduling of business activities usually refers to business months or weeks, follow-up visits are usually planned on working days, TFDs must allow one to express temporal constraints at different time granularities.

In this paper, we propose a general framework that makes it possible to formally specify TFDs, possibly involving multiple time granularities, and to check whether or not a given database instance satisfies them. We will prove that the proposed framework subsumes all existing formalisms for TFDs, and it allows one to express TFDs which are not captured by them. As an example, assuming months as the basic time unit, we can encode constraints such as "*employees with the same role, who will not change it from the current month to the next one, will get the same (unchanged) salary*". Moreover, we will show the effectiveness of the approach by applying it to a real-world medical domain, related to the administration of chemotherapies.

The paper is organized as follows. In Section 2, we provide the motivating scenario. In Section 3, we describe the basic features (i.e., the temporal data model and the temporal views), the proposed framework relies on. In Section 4, we introduce a new formalism for the representation of TFDs, and we show that it allows one to capture a large variety of TFDs. Then, in Section 5, we explain how to bring back the problem of checking whether a temporal relation satisfies a given TFD to the problem of checking whether the evaluation of a suitable algebraic expression returns the empty relation. We conclude the section with a short discussion of computational aspects.

In Section 6, we briefly surveys existing systems for TFDs and we compare them with the framework we propose. Finally, in Section 7, we provide an assessment of the work and we outline future research directions.

2 A motivating scenario from clinical medicine

To illustrate the relevance of properly expressing and carefully checking temporal constraints on data, we consider a real-world example taken from the domain of chemotherapies for oncology patients. Most health care institutions collect a large quantity of clinical information about patients and physicians' actions, such as therapies and surgeries, and 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 modeled, in order to be able to properly represent clinical data and to reason on them. In the considered case, oncology patients undergo several chemotherapy cycles. Each one can be repeated several times, and it typically includes the administration of several drugs to be assumed according to a predefined temporal pattern.

The problem of managing chemotherapy plans has been extensively studied by the clinical research. Chemotherapy plans are described and recommended in detail in several clinical practice guidelines. Let us consider the following chemotherapy recommendations related to FAC and CEF regimens [1, 7] for the treatment of breast cancer.

Example 1 *Recommended FAC and CEF regimens.*

> *FAC regimen: "*The recommended FAC regimen consists of 5-fluorouracil on days 1 and 8, and doxorubicin and cycloshosphamide on day 1. This is repeated every 21 days for 6 cycles*"* (that is, 6 cycles of 21 days each).
>
> *CEF regimen: "*The recommended CEF regimen consists of 14 days of oral cycloshosphamide, and intravenous injection of epirubicin and 5-fluorouracil on days 1 and 8. This is repeated every 28 days for 6 cycles.*"*

A relation schema *Patient* to be used for storing information about patients who underwent chemotherapies can be structured as follows. For each patient, we store the type of therapy, the patient's identifier, the blood group, the name of physician who prescribes the therapy, the assumed drugs with their quantities, and the specific assumption time by means of attributes *Chemo*, *PatId*, *BG*, *Phys*, *Drug*, *Qty*, and *VT*, respectively. We assume that attribute VT specifies the valid time of a tuple in term of days (from a given, implicit day taken as the origin of the time domain).

Table 1 shows a possible instance of the relation schema *Patient* describing chemotherapy treatments for patients with *PatId* 1, 2, and 3. As an example, according to the prescription of the FAC treatment reported in Example 1, the first

day of the cycle the patient has to assume the drugs *Flu*, *Dox*, and *Cyc*, i.e., drugs containing 5-fluorouracil, doxorubicin, and cyclophosphamide, respectively.

TUPLE#	***Chemo***	***PatId***	***BG***	***Phys***	***Drug***	***Qty***	***VT***
1	FAC	1	0+	Smith	Flu	500	1
2	FAC	1	0+	Hubbard	Dox	50	1
3	FAC	1	0+	Verdi	Cyc	500	1
4	FAC	1	0+	Smith	Flu	500	8
5	CEF	2	AB-	Verdi	Cyc	600	1
6	CEF	2	AB-	Hubbard	Flu	600	1
7	CEF	2	AB-	Hubbard	Epi	60	1
8	CEF	2	AB-	Smith	Cyc	600	2
9	CEF	2	AB-	Verdi	Cyc	500	3
	..	..	..	..	..	..	..
20	CEF	2	AB-	Verdi	Cyc	600	8
21	CEF	2	AB-	Hubbard	Flu	600	8
22	CEF	2	AB-	Hubbard	Epi	60	8
	..	..	..	..	..	..	..
33	CEF	3	AB-	Verdi	Cyc	550	1

Table 1: An instance s of the relation schema $Patient$, storing data about chemotherapy treatments.

According to the clinical meaning of data, several requirements can be imposed on this relation schema. They stem from both clinical/medical reasons and organizational rules of the medical unit managing the chemotherapy administration. In the following, we provide some typical requirements to be represented and managed by the database systems.

(1) *For any given drug, a patient may have at most one assumption per day*: such a requirement prevents any patient from having two or more assumptions of the same drug during the same day. As the administration of a drug within a chemotherapy has relevant side effects on the patient status, it is quite obvious. Nevertheless, checking it prevents possible data insertion errors.

(2) *For any chemotherapy, the quantities of a given drug prescribed to a patient on two days which are at most 14 days far away cannot be different*: for any patient, such a requirement forces drug quantities of chemotherapies to remain unchanged whenever the assumptions take place within 14 days (notice that the relation depicted in Table 1 violates it). According to the considered chemotherapies, this amounts to impose that drug quantities cannot change during a chemotherapy cycle (obviously, they can change if the chemotherapy changes).

(3) *For any chemotherapy, the quantities of a given drug administered by assumptions that take place during the same month cannot be different*: this requirement

states that, regardless of the patient, for any chemotherapy and any month, the administered quantity of a drug is always the same. A possible explanation of such a requirement is that there exists some form of synchronization among different administrations of the same chemotherapy, which forces changes in drug quantities to be done only when changes in month occur (for instance, to take into account different seasonal conditions).

(4) *For any pair of assumptions of the same drug by the same patient on two consecutive days, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption*: this requirement constrains the assumptions of a drug on two consecutive days by imposing different patients not to assume different quantities of the drug on the next day, if they assume the same quantity of it on the current day. A possible explanation of such a requirement is that, regardless of the patient and the chemotherapy, the physician must follow a predefined therapy plan, suggested by the clinical practice, for what concerns the quantities of drugs administrated on consecutive days.

(5) *For any pair of assumptions of the same drug prescribed by the same physician to the same patient on two consecutive days, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption*: this requirement slightly differs from the previous one as it further constrains the physician to be the same.

(6) *For any pair of consecutive assumptions of the same drug by the same patient within the same chemotherapy, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption*: such a requirement constrains consecutive assumptions of a drug by a patient within an assigned chemotherapy. As a general rule, time delays between consecutive assumptions may differ from one drug to another. As an example, oral cycloshosphamide in the CEF regimen is assumed daily for 14 days, while there is an interval of 7 days between two consecutive intravenous injections of epirubicin.

(7) *For any pair of consecutive assumptions of the same drug by the same patient within the same month, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption*: such a requirement imposes suitable constraints on consecutive drug assumptions only when they occur during the same month. We may assume the rationale of this requirement to be the same as that of Requirement (4).

(8) *For any given chemotherapy, the quantities of drugs that are assumed (by patients) with consecutive assumptions that take place one 7 days after the other cannot change*: for any chemotherapy, such a requirement basically imposes that (only) the quantities of drugs assumed by patients every 7 days cannot change.

3 Temporal data model and temporal views

In the next two sections, we will outline a general framework for the specification and verification of different kinds of TFD. As a preliminary step, in this section, we describe the temporal data model we rely on.

To represent TFDs, we will take advantage of a simple temporal relational data model based on the notion of *temporal relation*. We assume the time domain T to be isomorphic to the set of natural numbers with the usual ordering $(\mathbb{N}, \leq)$. Let U be a set of atemporal attributes and VT be a temporal attribute, called valid time attribute. A temporal relation r is a relation on a temporal relation schema R with attributes $U \cup \{VT\}$. We use the notation $\mathbf{att}(r)$ to denote the set of (atemporal and temporal) attributes of the relation schema R of r. Moreover, given a tuple $t \in r$ and an attribute $A \in R$, we denote by $t[A]$ the value that t assumes on A. The temporal attribute VT specifies the valid time of a tuple, and it takes its value over the time domain T, that is, $t[VT] \in T$.

In the following, we will make use of the *tuple relational calculus* [9] to define suitable temporal views on data, that will help us in specifying and analyzing TFDs without harming the simplicity of the basic temporal data model. In particular, these views will allow us to easily "move through time" in order to establish a connection between corresponding tuples valid at different time points. A special role will be played by the following two temporal views, respectively called *next* and *nexttuple*, that allow one to link tuples that satisfy a specific temporal relation in order to represent relevant cases of (temporal) evolution.

Given a temporal distance k, with $k \geq 1$, the view *next* allows one to join pairs of corresponding tuples at distance k (for $k = 1$, it joins pairs of consecutive corresponding tuples). More precisely, given a temporal relation schema R, with attributes $U \cup \{VT\}$, a temporal relation r on R, and a pair of tuples $t, t' \in r$, the application of the view *next* to r, denoted $\chi_Z^{r,k}$, with $Z \subseteq U$ and $k \geq 1$, joins t, t' if (and only if) $t[Z] = t'[Z]$ and $t'[VT] = t[VT] + k$. Temporal view *next* is formally defined as follows.

Definition 1 *Let R be a temporal relation schema with attributes $U \cup \{VT\}$, $W = U - Z$, $\overline{W}$ be obtained from W by replacing each attribute $A \in W$ by $\overline{A}$, r be a temporal relation on R, and $t, t' \in r$. The relation view $\chi_Z^{r,k}$, with schema $ZW\overline{W} \cup \{VT, \overline{VT}\}$, is defined as follows:*

$$\begin{aligned}\chi_Z^{r,k} \stackrel{def}{=} \{s \mid\ & \exists t, t'(r(t) \wedge r(t') \wedge t[Z] = t'[Z] \wedge t'[VT] = t[VT] + k \wedge \\ & s[Z] = t[Z] \wedge s[W] = t[W] \wedge s[\overline{W}] = t'[W] \wedge \\ & s[VT] = t[VT] \wedge s[\overline{VT}] = t'[VT])\}\end{aligned}$$

Hereinafter, when no confusion may arise, we will write χ_Z^r for $\chi_Z^{r,1}$. It is worth pointing out that, in the definition of $\chi_Z^{r,k}$, we make use of the non-standard (arith-

metic) selection condition $t'[VT] = t[VT] + k$. However, it is just syntactic sugar, as the expression for $\chi_Z^{r,k}$ can be turned into a standard relational calculus expression, as shown in detail in the appendix.

Temporal view *nexttuple* allows one to join pairs of consecutive tuples (with respect to the values they assume on attribute VT). More precisely, given a temporal relation schema R, with attributes $U \cup \{VT\}$, a temporal relation r on R, and a pair of tuples $t, t' \in r$, the application of the view *nexttuple* to r, denoted τ_Z^r, with $Z \subseteq U$, joins t, t' if (and only if) $t[Z] = t'[Z]$ and t' is the tuple immediately following t with respect to VT. Temporal view *nexttuple* is formally defined as follows.

Definition 2 *Let R be a temporal relation schema with attributes $U \cup \{VT\}$, $W = U - Z$, $\overline{W}$ be obtained from W by replacing each attribute $A \in W$ by $\overline{A}$, r be a temporal relation on R, and $t, t' \in r$. The relation τ_Z^r, with schema $ZW\overline{W} \cup \{VT, \overline{VT}\}$, is defined as follows:*

$$\begin{aligned}\tau_Z^r \stackrel{def}{=} \{u \mid\ & \exists t, t'(r(t) \wedge r(t') \wedge t[Z = t'[Z] \wedge u[Z] = t[Z] \wedge u[W] = t[W] \wedge \\ & u[\overline{W}] = t'[W] \wedge u[VT] = t[VT] \wedge u[\overline{VT}] = t'[VT] \wedge \\ & t[VT] < t'[VT] \wedge \neg\exists t''(r(t'') \wedge t[Z] = t''[Z] \wedge \\ & t[VT] < t''[VT] \wedge t''[VT] < t'[VT]))\}\end{aligned}$$

Temporal view *nexttuple* allows one to associate a tuple t of r with its first evolution t', that is, it associates t with t' if there exists no $t'' \in r$ such that $t''[Z] = t[Z]$ and $t[VT] < t''[VT] < t'[VT]$.

It is worth emphasizing that temporal views *next* and *nexttuple* make it possible to represent non-trivial aspects of the temporal evolution of data. More precisely, *next* takes into account the "synchronous" evolution of data, that is, tuples are joined with the corresponding ones which hold k time points later, while *nexttuple* models "asynchronous" data evolutions as it joins tuples with the corresponding consecutive ones, which may hold at different time points. As an example, let us consider the instance of the relation schema *Patient* given in Table 1. The view $\chi_{PatId,Phys}^{Patient}$ joins tuples which have the same values for $PatId$ and $Phys$ and are valid at time points t and $t + 1$, respectively. The view $\tau_{PatId,Phys}^{Patient}$ joins pairs of tuples such that they have the same values for $PatId$ and $Phys$ and there are no tuples with the same values for $PatId$ and $Phys$ which hold at some time point in between the valid times of the two tuples.

4 A uniform framework for TFDs

In this section, we propose an original formalism for the specification of TFDs. We first introduce its syntax and semantics; then, we show that it allows one to

express all TFDs dealt with by existing formalisms as well as to cope with new classes of TFDs.

Definition 3 *Let R be a temporal relation schema with attributes $U \cup \{VT\}$. A TFD is an expression of the following form:*

$$[E\text{-}Exp(R), t\text{-}Group]X \rightarrow Y,$$

where $E\text{-}Exp(R)$ is a relational expression on R, called evolution expression*, $t\text{-}Group$ is a mapping $\mathbb{N} \rightarrow 2^{\mathbb{N}}$, called* temporal grouping*, and $X \rightarrow Y$ is a functional dependency. We distinguish two types of TFD:*

1. *The schema of the expression $E\text{-}Exp(R)$ is $U \cup \{VT\}$. Then, $X, Y \subseteq U$.*
2. *The schema of the expression $E\text{-}Exp(R)$ is $U\overline{U} \cup \{VT, \overline{VT}\}$, Then $X, Y \subseteq U\overline{U}$ and, for each $A \in Y$, both $XA \cap U \neq \emptyset$ and $XA \cap \overline{U} \neq \emptyset$.*

Temporal grouping specifies how to group tuples, on the basis of the values they assume on temporal attribute VT (and on attribute $\overline{VT}$, if present), when $X \rightarrow Y$ is evaluated.

For the sake of simplicity, we confined ourselves to TFDs involving at most two consecutive database states. However, Definition 3 can be easily generalized to the case of tuple evolutions involving n consecutive states.

Evolution expressions $E\text{-}Exp(R)$ take advantage of temporal views to select those tuples, valid at different time points, that must be merged in order to track the evolution of domain objects over time.

In principle, there are no restrictions on the form that mapping $t\text{-}Group$ may assume. However, such a generality is not necessary from the point of view of applications. In the following, we will restrict ourselves to specific mappings, as those captured by Bettini et al.'s granularities [2] and Wjisen's time relations [12]. A time granularity G is a partition of the time domain in groups of indivisible, disjoint units, called *granules*. Classical examples of granularity are Day, Working-Day, Week, and Month. Various formalisms for representing and reasoning about time granularities have been proposed in the literature, including Granular Calendar Algebra by Ning et al. [8] and Ultimately Periodic Automata, by Bresolin et al. [3]. Wijsen's time relations are subsets of the set $\{(i, j) \mid i \in \mathbb{N},\ j \in \mathbb{N},\ i \leq j\}$. As an example, we can define a time relation that, for any $i \in \mathbb{N}$, collects all and only the pairs of time points (i, j) such that $j - i \leq k$, for some fixed k. Granularities can be recovered as special cases of time relations, called chronologies (it is not difficult to see that there exist quite natural time relations that cannot be expressed by means of granularities).

We constrain $t\text{-}Group$ to take one of the following two forms:

$$
\begin{array}{lll}
t\text{-}Group & \stackrel{\text{def}}{=} & \mathsf{G}(\mathsf{i}) \\
& \text{where} & \mathsf{G}(\mathsf{i}) \text{ is the i-th granule of granularity } \mathsf{G} \\
t\text{-}Group & \stackrel{\text{def}}{=} & \bigcup_{j=1}^{n}\{(i+\alpha_j)\} \text{ for some } n \geq 1 \\
& \text{where} & \alpha_1 = 0, \forall j \in [1,n]\ \alpha_j \in \mathbb{N}, \text{ and } \forall k \in [1,n-1]\ \alpha_k < \alpha_{k+1}
\end{array}
$$

In the first case, given a granularity G, $t\text{-}Group$ groups time points according to the granule $\mathsf{G}(\mathsf{i})$ they belong to. In the second case, $t\text{-}Group$ groups time points into an infinite number of intersecting finite sets. Each i-th group consists of the i-th time point plus other time points identified by their offset with respect to such a point.

As in the case of standard FDs, a TFD is a statement about admissible temporal relations on a temporal relation schema R. We say that a temporal relation r on the temporal relation schema R satisfies a TFD $[E\text{-}Exp(R), t\text{-}Group]X \rightarrow Y$ if it is not possible that the relation obtained from r by applying the expression $E\text{-}Exp(R)$ (hereinafter, the *evolution relation*) features two tuples t, t' such that (i) $t[X] = t'[X]$, (ii) $t[VT]$ and $t'[VT]$ (the same for $t[\overline{VT}]$ and $t'[\overline{VT}]$, if present) belong to the same temporal group, according to the mapping $t\text{-}Group$, and (iii) $t[Y] \neq t'[Y]$. This amount to say that the FD $X \rightarrow Y$ must be satisfied by each relation obtained from the evolution relation by selecting those tuples whose valid times belong to the same temporal group.

We partition the set of relevant TFDs into four classes.

- *Pure temporally grouping TFDs*.
  The evolution expression $E\text{-}Exp(R)$ returns the given temporal relation. Tuples are grouped on the basis of $t\text{-}Group$.

- *Pure temporally evolving TFDs*.
  The evolution expression $E\text{-}Exp(R)$ merges tuples modeling the evolution of a real-world object. There is no temporal grouping, that is, there is only one group collecting all tuples of the computed relation.

- *Temporally mixed TFDs*.
  First, the evolution expression $E\text{-}Exp(R)$ merges tuples modeling the evolution of a real-world object; then, temporal grouping is applied to the resulting tuples.

- *Temporally hybrid TFDs*.
  First, the evolution expression $E\text{-}Exp(R)$ selects those tuples of the given

temporal relation that contribute to the modeling of the evolution of a real-world object (that is, it removes isolated tuples); then, temporal grouping is applied to the resulting set of tuples.

In the following, we will describe in some detail the above classes of TFDs and we will give some meaningful examples of TFDs belonging to them.

Pure temporally grouping TFDs.
In these TFDs, $E\text{-}Exp$ returns the (original) temporal relation r. This force the FD $X \rightarrow Y$, with $X, Y \subseteq U$, to be checked on every (maximal) subset of r consisting of tuples whose VT values belong the same temporal group.

Example 2 *Let us consider the first three requirements of relation* Patient *reported in Section* 2*:*

1. for any given drug, a patient may have at most one assumption per day*;*
2. for any chemotherapy, the quantities of a given drug prescribed to a patient on two days which are at most 14 days far away cannot be different*;*
3. for any chemotherapy, the quantities of a given drug administered by assumptions that take place during the same month cannot be different.

The first requirement is captured by the following TFD:

$$[Patient, \{i\}]PatId, Drug \rightarrow Chemo, BG, Phys, Qty$$

For each time point i, the FD $PatId, Drug \rightarrow Chemo, BG, Phys, Qty$ must be satisfied by the tuples of (the instance of the relation schema) $Patient$ valid at time i. As a matter of fact, this forces attributes $PatId, Drug$ to be a *snapshot key* for relation schema $Patient$: the set of tuples of $Patient$ valid at a given time point (snapshot) must have $PatId, Drug$ as a (standard) key [5].

The second requirement can be encoded as follows:

$$[Patient, \{i, i+1, \ldots, i+13\}]PatId, Chemo, Drug \rightarrow Qty$$

In such a way, we force the FD $PatId, Chemo, Drug \rightarrow Qty$ to be checked at each time point i on the tuples of (the instance of the relation schema) $Patient$ valid at time points $i, i+1, i+2, \ldots$, or $i+13$, that is, for each i, $Chemo, PatId, Drug \rightarrow Qty$ must be satisfied by the tuples belonging to the union of snapshots of the temporal relation at time instants $i, \ldots, i+13$.

The third requirement can be expressed by means of the following TFD:

$$[Patient, \mathsf{Month(i)}]\, Chemo, Drug \rightarrow Qty$$

Pure temporally evolving TFDs.
In these TFDs, the evolution expression $E\text{-}Exp(R)$ returns a relation over the schema $U\overline{U} \cup \{VT, \overline{VT}\}$ which is computed by means of join operations on some attribute subset of U, while temporal grouping considers tuples valid at each time point in isolation.

Example 3 *Let us consider the following three requirements of relation* Patient *reported in Section* 2*:*

4. for any pair of assumptions of the same drug by the same patient on two consecutive days, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption*;*

5. for any pair of assumptions of the same drug prescribed by the same physician to the same patient on two consecutive days, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption*;*

6. For any pair of consecutive assumptions of the same drug by the same patient within the same chemotherapy, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption.

Let $\mathsf{Top(i)}$ be the top granularity collecting all time points in a single nonempty granule [2]. Requirement (4) can be formalized as follows:

$$[\chi^{Patient}_{PatId,Chemo}, \mathsf{Top(i)}]Drug, Qty \rightarrow \overline{Qty}$$

while Requirement (5) is captured by the following TFD:

$$[\chi^{Patient}_{PatId,Chemo,Phys}, \mathsf{Top(i)}]Drug, Qty \rightarrow \overline{Qty}$$

TFDs in this class can be viewed as a generalization of Vianu's dynamic dependencies [10]: the evolution expression allows one to define the evolution mapping (update mapping, according to Vianu's terminology) that associates each tuple valid at a time i with its corresponding tuple (if any) valid at time $i + 1$ taking advantage of the values of specific relation attributes. In the relational framework we propose, evolution mappings are thus expressed by means of suitable joins on a subset of U. Moreover, TFDs for Requirements (4) and (5) show the possibility of defining dynamic dependencies according to a number of evolution mappings.

Let us consider now Requirement (6). To cope with it, we need the ability to join tuples which are possibly not valid at consecutive time points. Consecutive assumptions of the same drug by the same patient may indeed occur on consecutive time points (this is the case with tuples #8 and #9 in Table 1), but they may

also take place on time points which are far away from one another (this is the case with tuples #1 and #4 in Table 1), and thus we must be able to deal with a kind of temporal asynchronous evolution. Requirement (6) is encoded by the following TFD:

$$[\tau^{Patient}_{PatId,Chemo,Drug}, \mathsf{Top(i)}]Drug, Qty \rightarrow \overline{Qty}$$

Temporally mixed TFDs.
In these TFDs, the evolution expression $E\text{-}Exp$ returns a relation over the schema $U\overline{U}\cup\{VT, \overline{VT}\}$ which is computed by means of join operations on some attribute subset of U, while temporal grouping groups together tuples according to their VT values. In such a way, one can define evolving (that is, dynamic) dependencies that must hold at all (and only) time points *belonging to the same temporal group*.

Example 4 *Let us now consider the seventh requirement of relation* Patient *reported in Section* 2*:*

7. For any pair of consecutive assumptions of the same drug by the same patient within the same month, the quantity of the second assumption uniquely depends on (the drug and) the quantity of the first assumption.

Such a requirement is encoded by the following TFD:

$$[\chi^{Patient}_{PatId,Drug}, \mathsf{Month(i)}]Drug, Qty \rightarrow \overline{Qty}$$

Temporally hybrid TFDs.
In these TFDs, the evolution expression $E\text{-}Exp$ returns a relation over the schema $U \cup \{VT\}$ computed by means of join operations on some attribute subset of U and further renaming, project, and union operations, while temporal grouping groups together tuples according to their VT values.

These TFDs allow one to express requirements on evolving values that must be preserved by all evolutions. As we will show in Section 6, such a class of requirements is captured neither by Vianu's dynamic dependencies nor by the other "grouping" TFDs proposed in the literature.

Example 5 *Let us now consider the last requirement of the relation* Patient *reported in Section* 2*:*

8. For any given chemotherapy, the quantities of drugs that are assumed (by patients) with consecutive assumptions that take place one 7 days after the other cannot change.

In this case, grouping tuples which "happen" every seven days and then checking dependency $Chemo, Drug \rightarrow Qty$ against them is not enough, and thus TFD $[r, \{i\} \cup \{i+7\}]Chemo, Drug \rightarrow Qty$ does not help us in representing this last requirement.

A TFD that takes into consideration the evolution of tuples in an appropriate way is the following one:

$$[He^{Patient}, \mathsf{Top(i)}]\, Chemo, Drug \rightarrow Qty,$$

where

$$\begin{aligned} He^{Patient} \quad \stackrel{\text{def}}{=} \{t| \quad & \exists t'(\tau^{Patient}_{PatId,Drug}(t') \wedge t'[\overline{VT}] = t'[VT] + 7 \wedge \\ & t[U] = t'[U] \wedge t[VT] = t'[VT]) \vee \\ & \exists t'(\tau^{Patient}_{PatId,Drug}(t') \wedge t'[\overline{VT}] = t'[VT] + 7 \wedge \\ & t[U] = t'[\overline{U}] \wedge t[VT] = t'[\overline{VT}])\}. \end{aligned}$$

The evolution expression $He^{Patient}$ first joins tuples representing consecutive administrations (of a given drug to a given patient), that is, pairs of administrations with no administrations in between. Then, two sets of tuples, respectively featuring the attribute values of the first and of the second drug administration, are computed by two existential subqueries. Finally, the two sets are merged (logical disjunction) to make it possible to specify (and check) the functional dependency. In such a way, the functional dependency is only checked on tuples referring to consecutive administrations of a given drug to a given patient that take place on time points (days) which are 7 unit (days) from one another. As all tuples have to be considered together, temporal grouping is $\mathsf{Top(i)}$.

We conclude the section by providing another example of this new kind of temporal functional dependency. Let us consider the constraint "*employees with the same role, who will not change it from the current month to the next one, will get the same (unchanged) salary*" that we already mentioned in the introduction. We need both to join consecutive tuples modeling old and new values for a given employee (this can be done with Vianu's DFDs as well) and to compare these old and new values (no one of existing formalisms for TFDs support these comparisons). Given the relation $Employee$ over the schema $U = \{empId, salary, role\}$, the above constraint can be encoded by means of the following temporally hybrid TFD:

$$[ev^{Employee}, \{i\} \cup \{i+1\}]role \rightarrow salary,$$

where

$$\begin{aligned} ev^{Employee} \quad \stackrel{\text{def}}{=} \{t| \quad & \exists t'(\chi^{Employee}_{empId,role}(t') \wedge t[U] = t'[U] \wedge t[VT] = t'[VT]) \vee \\ & \exists t'(\chi^{Employee}_{empId,role}(t') \wedge t[U] = t'[\overline{U}] \wedge t[VT] = t'[\overline{VT}])\}. \end{aligned}$$

The evolution expression $ev^{Employee}$ joins tuples related to the same employee with the same role holding at two consecutive time points, taking months as the basic time unit. Then, two sets of tuples, respectively featuring the attribute values holding at the first and the second time points, are computed by two existential subqueries. Finally, the two sets are merged (logical disjunction) to make it possible to specify (and check) the functional dependency $role \rightarrow salary$. In such a way, tuples describing employees that change their role and, in the new role, get a salary different from that of the other employees with the same role are allowed, as they do not appear in $ev^{Employee}$.

5 TFD checking

In this section, we show that the problem of checking whether a temporal relation r satisfies a given (set of) TFD(s) can be reduced to the problem of establishing whether the evaluation of a suitable relational query on r returns the empty relation [6]. To make the computational steps of the checking procedure explicit, we adopt the *named relational algebra* featuring selection, projection, natural join, set difference, set union, and renaming as its basic operations [6].

Any TFD on a relation schema R can be viewed as an instance of the following general pattern:

$$[E\text{-}Exp(R), t\text{-}Group]X \rightarrow Y$$

We can verify whether a given relation R satisfies the TFD by checking the emptiness of the result q of the following expression:

$$q \leftarrow \sigma_{Cnd}(E\text{-}Exp(R) \bowtie_{X=\widehat{X}} \rho_{W \rightarrow \widehat{W}} E\text{-}Exp(R))$$

where W is the set of attributes of $E\text{-}Exp(R)$ and Cnd stands for $\bigvee_{A \in Y}(A \neq \widehat{A}) \wedge SameTGroup$.

The predicate $SameTGroup$ verifies that all the given valid times belong to the same group, according to the expression $t\text{-}Group$. Thus, if $E\text{-}Exp(R)$ is defined over the schema $U\overline{U} \cup \{VT, \overline{VT}\}$, then $SameTGroup \equiv \exists\, i(VT \in t\text{-}Group(i) \wedge \overline{VT} \in t\text{-}Group(i) \wedge \widehat{VT} \in t\text{-}Group(i) \wedge \widehat{\overline{VT}} \in t\text{-}Group(i))$; if $E\text{-}Exp(R)$ is defined over the schema $U \cup \{VT\}$, then $SameTGroup \equiv \exists\, i(VT \in t\text{-}Group(i) \wedge \widehat{VT} \in t\text{-}Group(i))$.

Let us now focus on the two different ways of specifying the temporal grouping, considering, for sake of simplicity, the general case of the evolution relation defined on the schema $U \cup \{VT\}$. The predicate $SameTGroup$ has to deal either with a granularity specification or with intersecting finite sets of time points.

When dealing with intersecting finite sets of time points, the predicate consists of a disjunction of conditions on the relative distances between the involved valid times. More precisely, if $t\text{-}Group = \bigcup_{j=1}^{n}\{(i+\alpha_j)\}$, then

$$SameTGroup \stackrel{\text{def}}{=} \bigvee_{j=1}^{n}(\widehat{VT} = VT + \alpha_j)$$

In the case of a granularity specification, as we will focus on checking temporal functional dependencies on any (finite) database, we will assume to deal with some suitable finite set of granules of a given granularity, that is, those granules containing some valid tuples. In particular, the considered portion of granularity G is described by means of a temporal relation $Gran$ defined on the attributes (G_Id, I, G_s, G_e), where tuples represent the name of the granularity (G_Id), the index (I), the starting point (G_s), and the ending point (G_e) of each granule of the considered granularity. For sake of simplicity, we restrict ourselves to granularities without gaps inside.

In this case, when $t\text{-}Group$ stands for $G(i)$ (let *"G"* be the granularity name in its relational representation), the problem of checking whether the TFD is satisfiable is equivalent to checking whether the following relational algebra query q returns the empty set:

$q \leftarrow \sigma_{Cnd}((E\text{-}Exp(R) \bowtie_{X=\widehat{X}} (\rho_{W\rightarrow\widehat{W}} E\text{-}Exp(R))) \bowtie Gran)$

where $Cnd \equiv \vee_{A\in Y}(A \neq \widehat{A}) \wedge VT \geq G_s \wedge VT \leq G_e \wedge \widehat{VT} \geq G_s \wedge \widehat{VT} \leq G_e \wedge G\_Id =$ "G".

It is quite straightforward to prove that checking for emptiness the resulting relation q is equivalent to say that the given relation R satisfy the considered TFD.

Let us now discuss in detail the checking-for-emptiness approach for each type of the previously introduced TFDs by providing some examples for each kind of TFD.

Pure temporally grouping TFDs. The first dependency, which is related to Requirement (1), can be expressed as follows: $[Patient, \{i\}]PatId, Drug \rightarrow Chemo, BG, Phys, Qty$. The check for emptiness on a given relation $Patient$ can be performed by executing the query q:

$q \leftarrow \sigma_{Cnd}(Patient \bowtie_{PatId=\widehat{PatId} \wedge Drug=\widehat{Drug}} \rho_{W\rightarrow\widehat{W}} Patient))$,

where $Cnd \equiv (Chemo \neq \widehat{Chemo} \vee BG \neq \widehat{BG} \vee Phys \neq \widehat{Phys} \vee Qty \neq \widehat{Qty}) \wedge VT = \widehat{VT}$.

The second dependency, which is related to Requirement (2), can be represented as $[Patient, \{i\} \cup .. \cup \{i+13\}]Chemo, PatId, Drug \rightarrow Qty$. We can establish whether $Patient$ satisfies it by checking for emptiness the following query q:

$$q \leftarrow \sigma_{Qty \neq \widehat{Qty} \wedge (\widehat{VT} - VT) \leq 13}(Patient \bowtie_{Chemo=\widehat{Chemo} \wedge PatId=\widehat{PatId} \wedge Drug=\widehat{Drug}} \rho_{W \rightarrow \widehat{W}} Patient).$$

The third TFD, which is exemplified by Requirement (3), can be expressed as $[Patient, \mathsf{Month(i)}]Chemo, Drug \rightarrow Qty$. It can be verified by taking advantage of the following query:

$$q \leftarrow \sigma_{Cnd}(Patient \bowtie_{Chemo=\widehat{Chemo} \wedge Drug=\widehat{Drug}} \rho_{W \rightarrow \widehat{W}} Patient) \bowtie Gran),$$

where $Cnd \equiv Qty \neq \widehat{Qty} \wedge VT \geq G_s \wedge VT \leq G_e \wedge \widehat{VT} \geq G_s \wedge \widehat{VT} \leq G_e \wedge G\_Id = \text{“}Month\text{”}$.

Pure temporally evolving TFDs. Requirement (4) can be expressed by means of the TFD $[\chi^{Patient}_{PatId,Chemo}, \mathsf{Top(i)}]Drug, Qty \rightarrow \overline{Qty}$ which can be checked for emptiness by means of the query:

$$q \leftarrow \sigma_{\overline{Qty} \neq \widehat{\overline{Qty}}}(\chi^{Patient}_{PatId,Chemo} \bowtie_{Drug=\widehat{Drug} \wedge Qty=\widehat{Qty}} (\rho_{W \rightarrow \widehat{W}} \chi^{Patient}_{PatId,Chemo})).$$

Similarly, the fifth TFD $[\chi^{Patient}_{PatId,Chemo,Phys}, \mathsf{Top(i)}]Drug, Qty \rightarrow \overline{Qty}$ can be verified on a relation s by checking for emptiness the following query:

$$q \leftarrow \sigma_{\overline{Qty} \neq \widehat{\overline{Qty}}}(\chi^{Patient}_{PatId,Chemo,Phys} \bowtie_{Drug=\widehat{Drug} \wedge Qty=\widehat{Qty}} \rho_{W \rightarrow \widehat{W}} \chi^{Patient}_{PatId,Chemo,Phys})$$

Requirement (6) can be expressed by means of the TFD:

$[\tau^{Patient}_{PatId,Chemo,Drug}, \{i\}]Drug, Qty \rightarrow \overline{Qty}$,

which can be checked for emptiness through the query:

$$q \leftarrow \sigma_{\overline{Qty} \neq \widehat{\overline{Qty}} \wedge VT = \widehat{VT}}(\tau^{Patient}_{PatId,Chemo,Drug} \bowtie_{Drug=\widehat{Drug} \wedge Qty=\widehat{Qty}} \rho_{W \rightarrow \widehat{W}} \tau^{Patient}_{PatId,Chemo,Drug})$$

Temporally mixed TFDs. Requirement (7) can be encoded by means of the TFD $[\chi^{PatientPatId,Drug}_{,}\mathsf{Month(i)}]Drug, Qty \rightarrow \overline{Qty}$, which can be verified by checking for emptiness the following query q:

$$q \leftarrow \sigma_{\overline{Qty} \neq \widehat{\overline{Qty}} \wedge VT \geq G_s \wedge VT \leq G_e \wedge \widehat{VT} \geq G_s \wedge \widehat{VT} \leq G_e}(\chi^{Patient}_{PatId,Drug} \bowtie_{Drug=\widehat{Drug} \wedge Qty=\widehat{Qty}} \rho_{W \rightarrow \widehat{W}} \chi^{Patient}_{PatId,Drug} \bowtie \sigma_{G\_Id=\text{“}Month\text{”}} Gran)$$

Temporally hybrid TFDs. Finally, Requirement (8) of Example 5 can be encoded by means of the TFD $[He^{Patient}, \mathsf{Top(i)}]Chemo, Drug \rightarrow Qty$. The corresponding query to check for emptiness is:

$$q \leftarrow \sigma_{Qty \neq \widehat{Qty}}(He^{Patient} \bowtie_{Drug=\widehat{Drug} \wedge Chemo=\widehat{Chemo}} \rho_{W \rightarrow \widehat{W}} He^{Patient})),$$

which can be expanded as follows:

$$q \leftarrow \sigma_{Qty \neq \widehat{Qty}}((\pi_{U \cup \{VT\}}(\sigma_{\overline{VT}=VT+7}(\tau^{Patient}_{PatId,Drug})) \bigcup \rho_{\overline{U},\overline{VT} \rightarrow U,VT} \pi_{\overline{U} \cup \{\overline{VT}\}}(\sigma_{\overline{VT}=VT+7}(\tau^{Patient}_{PatId,Drug}))) \bowtie_{Drug=\widehat{Drug} \wedge Chemo=\widehat{Chemo}}$$

$(\pi_{U\cup\{VT\}}(\sigma_{\overline{VT}=VT+7}(\tau^{Patient}_{PatId,Drug})) \bigcup$
$\rho_{\overline{U},\overline{VT}\to U,VT}\pi_{\overline{U}\cup\{\overline{VT}\}}(\sigma_{\overline{VT}=VT+7}(\tau^{Patient}_{PatId,Drug}))))$

We conclude the section with a short analysis of the computational complexity of TFD checking. As TFDs are represented by general expressions of the form:

$$[E\text{-}Exp(R), t\text{-}Group]X \to Y$$

the computational cost of checking for the satisfaction of $X \to Y$ on the evolution relation, with respect to the grouping condition $t\text{-}Group$, mainly depends on the structure of the $E\text{-}Exp(R)$ expression itself.

Let n and n_E be the cardinalities of the relation r and of the evolution relation $E\text{-}Exp(R)$, respectively. The emptiness check consists of the execution of the join:

$$E\text{-}Exp(R) \bowtie_{X=\widehat{X}} E\text{-}Exp(R),$$

followed by the application of a selection to the resulting relation to verify the condition on the consequents and on the temporal grouping. In the worst case, the join reduces to a cartesian product, that can be computed in $O(n_E^2)$. Then, the subsequent selection operation must be executed n_E^2 times (as many times as the tuples of the resulting relation are). Hence, the overall cost of the emptiness check is $O(n_E^2)$. As the number n_E ranges from n (for pure temporally grouping TFDs) to $O(n^2)$ (for temporally evolving relations), the complexity of the emptiness check ranges from $O(n^2)$, for pure temporally grouping relations, to $O(n^4)$, for temporally evolving relations.

6 Related work

Various representation formalisms for TFDs have been developed in the literature [2, 5, 10, 11, 12], which differ a lot in their structure as well as in the underlying data model. All of them basically propose alternative extensions to the relational model, often introducing non-relational features (this is the case with Wijsen's objects [12] and Vianu's update mappings [10]), making it difficult to identify their distinctive features and to systematically compare them in order to precisely evaluate their relative strength and their limitations. In the following, we take the set of requirements given in Section 2 as a sort of benchmark for their evaluation. A systematic analysis is provided in the appendix, where we first describe the most significant TFD formalisms proposed in the literature, following as much as possible the original formulation given by the authors, and, then, we formally prove that our proposal actually subsumes all of them. In the following, we provide a short account of such an analysis. In particular, we show that existing formalisms significantly differ in the requirements they are able to express

and that there exist meaningful requirements they are not able to cope with. As an example, existing TFD systems are not able to express Requirements (5-7), which (from the point of view of the conditions they impose) look like minor variations of Requirement (4).

Let us assume to have a representation of the patient database example in (the data model underlying) all the TFD systems we are going to analyze. We first show how to represent Requirements (1-4). As a matter of fact, not all these requirements can be encoded in all TFD systems.

In [5] Jensen et al. propose a bitemporal data model that allows one to associate both valid and transaction times with data. Jensen et al.'s TFDs make it possible to express conditions that must be satisfied at any (valid) time point taken in isolation. Requirement (1), which prevents any patient from having two or more assumptions of the same drug during the same day, can be modeled by Jensen et al. 's TFDs as follows:

$PatId, Drug \rightarrow^T Phys, Qty$

A general formalism for TFDs on complex (temporal) objects has been proposed by Wijsen in [12]. 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. Wijsen's TFDs have the form $c : X \rightarrow_\alpha Y$. Their meaning can be intuitively explained as follows. Let t_1 and t_2 be two objects of class c valid 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.

For any patient, Requirement (2) forces drug quantities of chemotherapies to remain unchanged whenever the assumptions take place within 14 days (the relation depicted in Table 1 violates such a requirement). Such a requirement constrains the duration of the time span between two assumptions and it can be modeled using Wijsen's TFDs as follows:

$Patient : PatId, Chemo, Drug \rightarrow_{14Days} Qty,$

where $14Days$ is a time relation grouping 14 consecutive days.

Bettini, Jajodia, and Wang's notion of TFD takes advantage of time granularity [2]. Their TFDs allow one to specify conditions on tuples associated with granules of a given granularity and grouped according to a coarser granularity. It is not difficult to show that Wijsen's TFDs actually subsumes Bettini et al.'s TFDs. More precisely, Bettini et al.'s TFDs are exactly all and only Wijsen's TFDs on chronologies (the class TFD-C in Wijsen's terminology).

Requirement (3) essentially states that, regardless of the patient, for any chemotherapy and any given month, the administered quantity of a drug is always the

same. This requirement can be expressed in Bettini et al.'s formalism for TFDs by the following TFD on the temporal module schema $(Patient, \mathsf{Day})$:

$Chemo, Drug \rightarrow_{\mathsf{Month}} Qty$

where Month is the granularity grouping days of the same month.

Its representation in Wijsen's TFD formalism is as follows:

$Patient : Chemo, Drug \rightarrow_{Month} Qty,$

where $Month$ is a time relation grouping days of the same month. It is not difficult to show that pure temporally grouping TFDs are very close to the TFDs proposed by Jensen et al. , Wijsen, and Bettini et al. , as witnessed by the above three temporal functional dependencies.

In [10], Vianu proposes a simple extension to the relational model in order to describe the evolution of a database over time. He defines a *database sequence* as a sequence of consecutive instances of the database, plus "*update mappings*" from one instance (the "old" one) to the next one (the "new" instance). Constraints on the evolution of attribute values of tuples (objects) over time are expressed by means of dynamic functional dependencies (DFDs), that make it possible to define dependencies between old and new values of attributes on updates. For example, Requirement (4) constrains the assumptions of a drug on two consecutive days by imposing different patients not to assume different quantities of the drug on the next day, if they assume the same quantity of it on the current day. Assuming that the update mapping represents the evolution of a tuple for a given patient and a given drug, Requirement (4) can be expressed by the following DFD, where for each attribute A, $\overset{\vee}{A}$ represents its old value and $\hat{A}$ its new value:

$\overset{\vee}{Drug}, \overset{\vee}{Qty} \rightarrow \overset{\wedge}{Qty}$

We conclude the section by an intuitive account of the fact that the last four requirements cannot be dealt with by existing TFD systems. This is true, in particular, for Requirements (5-7) that present some similarities with Requirement (4), which can be easily encoded using Vianu's DFDs. Consider, for instance, Requirement (5). Such a requirement is based on an update mapping which differs from the one of Requirement (4) in an essential way. Such a mapping must indeed associate any tuple involving a patient, a drug, and a physician, valid on a given day, with a tuple involving the same patient, drug, and physician, valid on the next day (if any). Unfortunately, Vianu's DFDs cannot help, as they are based on a fixed update mapping (that does not allow one to constrain the physician to be the same). Moreover, Vianu's update mappings are partial one-to-one mappings, and thus they cannot be exploited to deal with the case where a tuple, valid on a given day, must be associated with more than one tuple, valid on the next day. Requirement (6) constrains consecutive assumptions of a drug possibly involving

different delays: Vianu's update mappings cannot cope with asynchronous updates of different tuples. Requirement (7) cannot be fulfilled by existing TFDs as well. It indeed requires a sort of combination of Vianu's DFDs and tuple temporal grouping supported by Wijsen's TFDs (and by Bettini et al.'s TFDs).

Requirement (8) deserves a deeper analysis. Basically, it imposes that, for any chemotherapy, (only) the quantities of drugs assumed by patients every 7 days cannot change. On the one hand, this constraint cannot be expressed via Vianu's DFDs, as they do not allow one to formulate a condition of the form: "something cannot change in the evolution of the database". On the other hand, the other TFD systems have no the capability of "mapping tuples' evolution" (the evolution of a database can only be modeled through the union of consecutive states). As an example, Wijsen's TFD $Patient : Chemo, Drug \rightarrow_{7Days} Qty$, where the time relation $7Days$ groups days which are exactly 7 days far from each other, does not capture the intended meaning. Consider, for instance, the tuples belonging to the relation depicted in Table 1. The relation violates such a TFD as the tuple for patient 3 at time 1 violates it with respect to drug *Cyc* and chemotherapy *CEF*, with respect to time points 1 and 8. On the contrary, according to its intended meaning, requirement (8) is actually fulfilled by the relation depicted in Table 1, as the drugs that are assumed by patients every 7 days (*Flu* by patient 1, and *Epi* by patient 2) do not change their quantities from one assumption to the successive one (7 days later).

7 Conclusions and Future Work

In this paper, we focused our attention on the specification and checking of temporal functional dependencies (TFDs), possibly involving multiple time granularities. To overcome the limitations of existing TFDs, we have proposed a new *general notion* of TFD, that subsumes all of them and allows one to cope with temporal requirements they cannot deal with. The simplest TFDs are directly brought back to atemporal FDs; to manage the most complex ones some additional machinery is needed. As for the problem of checking whether a temporal relation satisfies a given set of TFDs, we have shown how to uniformly reduce it to the problem of checking for emptiness a suitable relational algebra expression over the considered temporal relation.

References

[1] V. Assikis, A. Buzdar, Y. Yang, and et al. A phase iii trial of sequential adjuvant chemotherapy for operable breast carcinoma: final analysis with

10-year follow-up. *Cancer*, 97:2716–2723, 2003.

[2] C. Bettini, S. Jajodia, and X. Wang. *Time granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, 2000.

[3] D. Bresolin, A. Montanari, and G. Puppis. A theory of ultimately periodic languages and automata with an application to time granularity. *Acta Informatica*, 46(5):331–360, 2009.

[4] J. Clifford and D. Warren. Formal semantics for time in databases. *ACM Transaction on Database Systems*, 8(2):214–254, 1983.

[5] C. Jensen, R. Snodgrass, and M. Soo. Extending existing dependency theory to temporal databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):563–581, 1996.

[6] P. C. Kanellakis. Elements of relational database theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, pages 1073–1156. Elsevier and MIT Press, 1990.

[7] M. Levine, C. Sawka, and D. Bowman. Clinical practice guidelines for the care and treatment of breast cancer: 8. Adjuvant systemic therapy for women with node-positive breast cancer (2001 update). *Canadian Medical Association journal*, page 164, 2001.

[8] P. Ning, S. Jajodia, and X. S. Wang. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36:5–38, 2002.

[9] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.

[10] V. Vianu. Dynamic functional dependency and database aging. *Journal of the ACM*, 34(1):28–59, 1987.

[11] J. Wijsen. Design of temporal relational databases based on dynamic and temporal functional dependencies. In J. Clifford and A. Tuzhilin, editors, *International Workshop on Temporal Databases*, Recent Advances in Temporal Databases, pages 61–76. Springer, 1995.

[12] J. Wijsen. Temporal FDs on complex objects. *ACM Transactions on Database Systems*, 24(1):127–176, 1999.

[13] J. Wijsen. Temporal dependencies. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 2960–2966. Springer US, 2009.

A Dealing with extended relations

As the additional relational views apparently use arithmetic operations, we will introduce an extended representation of temporal relations making it evident that arithmetic expressions occurring in the definition of temporal relational views are just shorthands (connected to the chosen representation for relations with a temporal dimension).

We call *snapshot* of the temporal relation r at time i, denoted r_i, the (atemporal) relation containing all and only the tuples of r valid at time point i. Formally, the snapshot of r at time i is defined as follows:

$$r_i \quad \stackrel{\text{def}}{=} \{t| \quad r(t) \wedge t.VT = i\}$$

The *extended temporal relation* r^E obtained from a temporal relation r is a temporal relation defined on the set of attributes

$$\mathbf{att}(r^E) = \mathbf{att}(r) \cup \{PRES?\} = U \cup \{VT, PRES?\},$$

where $PRES?$ is a Boolean attribute denoting the presence or absence of a tuple in the relation r at a given time point.

The extension of a temporal relation r is obtained as follows: for each time point i, if $t \in r_i$ (that is, if $t[VT] = i$), then a tuple t^E is inserted in r^E with values:

$$\begin{aligned} t^E[VT] &= i \\ t^E[A] &= t[A] \qquad \forall A \in U \\ t^E[PRES?] &= true \end{aligned}$$

Moreover, for each time point i between the earliest and the latest time points were some tuple of r holds, and each tuple t, with $t \in r_j$ (for some $j \neq i$) and $t \notin r_i$, a tuple s^E is inserted in r^E with values:

$$\begin{aligned} s^E[VT] &= i \\ s^E[A] &= t[A] \qquad \forall A \in U \\ s^E[PRES?] &= false \end{aligned}$$

r^E can be derived through the following query:

$$\begin{aligned} r^E \quad = \{t^E \quad | \quad & \exists t(r(t) \wedge t^E.VT = t.VT \wedge t^E.U = t.U \wedge t^E.PRES?) \vee \\ & \exists t(r(t) \wedge \exists t', t''(r(t') \wedge r(t'') \wedge \neg\exists t'''(r(t''') \wedge t'''.VT < t'.VT) \wedge \\ & \neg\exists t'^v(r(t'^v) \wedge t'^v.VT > t''.VT) \wedge t^E.VT \geq t'.VT \wedge \\ & t^E.VT \leq t''.VT) \wedge \neg\exists t^v(r(t^v) \wedge t^v.VT = t^E.VT \wedge t.U = t^v.U) \\ & \wedge\neg\ t^E.PRES? \wedge t^E.U = t.U)\} \end{aligned}$$

Thus, for every time point i the snapshot of an extended temporal relation at time point i is defined as follows:

$$r_i^E \ = \{t^E \mid \ r^E(t^E) \wedge t^E.VT = i\}$$

The notion of extended temporal relation resembles that of *completed relation* introduced by Clifford and Warren in [4]. This notion of *completed relation* has been exploited to complete a relation by adding to its temporal schema a new Boolean attribute that expresses the presence or absence of tuples (objects) at a given time point. In the original proposal, such a Boolean attribute, called EXISTS?, is added to the schema of the considered relation to indicate which tuples are of interest in any state. In those states where a given tuple does not exist, the attribute EXISTS? assumes value 0 and all the other attributes, but key ones, assume value $\perp$, a distinguished value whose meaning is that the attribute does not apply. Such an extension allows one to follow tuples (objects) throughout all of the states of the database. To this end, it is necessary to first collect all the tuples/objects (identified by their key values) that are present in some state of the database, and then to add the appropriate "*n*ull tuples" to states where no information about some tuples/objects is given. Our notion of extended temporal relation can be viewed as a generalization of the concept of completed relation, when no key attributes exist (or are defined) for (each state of) the considered temporal relation.

In the definition of $\chi_Z^{r,k}$, we make use of the non-standard (arithmetic) selection condition $t'.VT = t.VT + k$. However, this is simply syntactic sugar, as $\chi_Z^{r,k}$ can be turned into a standard relational calculus expression on the extended relation r^E by the following expansion.

For $k \geq 1$, let m_k be inductively defined as follows:

$$\begin{aligned} m_k \ = \{u \mid \ & \exists t, t'(m_{k-1}(t) \wedge r^E(t') \wedge u.Z = t.Z \wedge u.Z = t'.Z \\ & \wedge u.X = t.X \wedge u.X^1 = t.X^1 \wedge \ldots \wedge u.X^{k-1} = t.X^{k-1} \wedge \\ & u.VT = t.VT \wedge u.VT^1 = t.VT^1 \wedge \ldots \wedge u.VT^{k-1} = t.VT^{k-1} \wedge \\ & u.PRES? = t.PRES? \wedge u.PRES?^1 = t.PRES?^1 \wedge \ldots \wedge \\ & \quad u.PRES?^{k-1} = t.PRES?^{k-1} \wedge \\ & u.X^k = t'.X \wedge u.VT^k = t'.VT \wedge u.PRES?^k = t'.PRES? \wedge \\ & \neg\exists t''(r^E(t'') \wedge t''.Z = t.Z \wedge t''.VT > t.VT^{k-1} \wedge t''.VT < t'.VT))\} \end{aligned}$$

while m_0 is r^E.

As it can be observed from the above expression, m_k is defined on the schema $ZXX^1X^2 \ldots X^kVT\ VT^1\ VT^2 \ldots VT^k PRES?\ PRES?^1\ PRES?^2 \ldots PRES?^k$, where there are k suitable renominations of attributes $PRES?$, VT and of attributes in set X.

Taking advantage of this inductive definition, for any fixed $k \geq 1$, we can define the *next* view $\chi_Z^{r,k}$ on the schema $X\overline{X}, VT\overline{VT}$ as follows:

$$\begin{aligned}\chi_Z^{r,k} \quad \stackrel{\text{def}}{=} \{u \mid\ & \exists t(m_k(t) \wedge u.X = m_k.X \wedge u.\overline{X} = m_k.X^k \wedge \\ & u.Z = m_k.Z \wedge u.VT = m_k.VT \wedge u.\overline{VT} = m_k.VT^k \\ & \wedge m_k.PRES? \wedge m_k.PRES?^k)\}\end{aligned}$$

The *next* view is built in such a way that at the first step ($k = 1$), the relation r^E is joined with a suitably renamed version of itself and then only tuples whose (renamed) valid time follows the first one are selected. The resulting relation (featuring the attributes of the extended relation and the corresponding renamed ones) is then joined with a suitably renamed version of it. By the nested query (in m_1), we are able to join each tuple in r^E with the corresponding one valid at the next time point. m_1 is then suitably joined to r^E to build m_2, and so on, according to above inductive definition.

B The main proposals for TFDs

In this section, we provide a short overview of the main formalisms for TFDs (provided with/ devoid of time granularity) proposed in the literature.

Jensen, Snodgrass, and Soo's TFDs. In [5] Jensen et al. propose a bitemporal data model that allows one to associate both valid and transaction times with data. The domains of valid and transaction times are the finite sets $\mathcal{D}_{VT}$ and $\mathcal{D}_{TT}$, respectively. A valid time chronon c^v is a time point belonging to $\mathcal{D}_{VT}$ and a transaction time chronon c^t is a time point belonging to $\mathcal{D}_{TT}$. A bitemporal chronon $c^b = (c^t, c^v)$ is an ordered pair consisting of a transaction time chronon and a valid time chronon. The schema of a bitemporal relation R, defined on the set $U = \{A_1, A_2, \ldots, A_n\}$ of atemporal attributes (called non-timestamp attributes), is of the form $R = (A_1, A_2, \ldots, A_n|T)$, that is, it consists of n atemporal attributes $A_1, A_2, \ldots, A_n$, with domain $dom(A_i)$ for each $i \in [1, n]$, and an implicit timestamp attribute T. The domain of T is $2^{(\mathcal{D}_{TT} \cup \{UC\}) \times \mathcal{D}_{VT}}$, where UC is a special value that can be assumed by a transaction time chronon to express the condition "until changed". For instance, to state that a tuple valid at time c^v is current in the database, the bitemporal chronon (UC, c^v) must be assigned to the tuple timestamp. As a general rule, they associate a set of bitemporal chronons in

the two-dimensional space with every tuple. Such a set, denoted by t^b, is called *bitemporal element*.

TFDs are FDs that must be satisfied at any bitemporal chronon (c^t, c^v) by tuples valid at time c^v and current at time c^t. Formally, TFDs are defined as follows.

Definition 4 *Let $R^B = R(U|T)$ be a temporal relation schema and $X, Y \subseteq U$. A database instance r^B of R^B satisfies a TFD $X \rightarrow^T Y$ iff*

$$\forall c^t \in \mathcal{D}_{TT} \cup \{UC\}\ \forall c^v \in \mathcal{D}_{VT}\ \forall s_1, s_2 \in \tau^V_{c^v}(\rho^B_{c^t}(r^B)) \\ (s_1[X] = s_2[X] \Rightarrow s_1[Y] = s_2[Y]),$$

where the expression $\tau^V_{c^v}(\rho^B_{c^t}(r^B))$ returns the set of tuples in r^B valid at c^v and current at c^t.

Jensen et al.'s TFDs make it possible to express conditions that must be satisfied at any (valid) time point taken in isolation. As an example, let *Emp* be a temporal relation schema with the set of atemporal attributes $U = \{empId, salary, role\}$. The condition "*at any time, the salary of an employee uniquely depends on his role*" can be expressed by the TFD $role \rightarrow^T salary$.

Bettini, Jajodia, and Wang's TFDs. Bettini, Jajodia, and Wang's notion of TFD takes advantage of time granularity [2]. A time granularity is a partition of a time domain in groups of indivisible units called *granules*. Examples of granularities are Day, Month, and WorkingDay. A time granularity can be formally defined as follows.

Definition 5 *A time granularity is a mapping G from integers to subsets of a totally ordered time domain $(T, <)$ such that: (1) if $i < j$ and $G(i), G(j) \neq \emptyset$, then, for all $n \in G(i)$ and $m \in G(j)$, $n < m$; (2) if $i < k < j$ and $G(i), G(j) \neq \emptyset$, then $G(k) \neq \emptyset$.*

The domain of a granularity G is called *index set* and the elements of its range are called *granules*. The *image* of a time granularity is the union of the granules in the granularity. A time granularity is associated with each relation schema in the database according to the following definition.

Definition 6 *A* temporal module schema *is a pair (R, G), where R is a relation schema and G is a time granularity. A* temporal module *is a triple (R, G, ϕ), where (R, G) is a temporal module schema and ϕ is a function, called* time windowing function*, that associates a set of tuples with every granule $G(i)$ (the set of tuples valid at $G(i)$).*

Let $G(i_1, \ldots, i_k)$ be a shorthand for $\bigcup_{1 \leq j \leq k} G(i_j)$. Bettini, Jajodia, and Wang's TFDs are defined as follows.

Definition 7 *Let X and Y be two (finite) sets of attributes and H be a time granularity such that $H(i) \neq \emptyset$ for some i. A TFD $X \rightarrow_H Y$ is satisfied by a temporal module $M = (R, G, \phi)$ if and only if for all tuples t_1, t_2 and all positive integers i_1, i_2, if (i) $t_1[X] = t_2[X]$, (ii) $t_1 \in \phi(i_1)$ and $t_2 \in \phi(i_2)$, and (iii) there exists j such that $G(i_1, i_2) \subseteq H(j)$, then $t_1[Y] = t_2[Y]$.*

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. As an example, if we consider the temporal module schema $(Emp,$ Month$)$, where Emp is a relation schema with attributes $U = \{empId,\ salary,\ role\}$ and Month is the granularity that groups time points of the same month, the condition "*for any given year, employees with the same role have the same salary; however, their salary may change from one year to the next one*" is captured by the TFD $role \rightarrow_{\mathsf{Year}} salary$.

Wijsen's TFDs. A general formalism for TFDs on complex (temporal) objects has been proposed by Wijsen in [12]. 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 $\mathbb{N} \otimes \mathbb{N}$, where the operation $\otimes$ is defined as $I \otimes J = \left\{ (i,j) \,\middle|\, i \in I \wedge j \in J \wedge i \leq j \right\}$, with $I, J \subseteq \mathbb{N}$. Examples of meaningful time relations are $Forever = \mathbb{N} \otimes \mathbb{N}$, $Next = \left\{ (i,j) \in Forever \,\middle|\, j - i \leq 1 \right\}$, and $Current = \left\{ (i,i) \,\middle|\, i \in \mathbb{N} \right\}$. 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's temporal data model is formally defined as follows. Let $\mathbf{dom}$ be a set of atomic values, that is, the union of disjoint domains corresponding to *atomic types*, $\mathbf{att}$ be a set of attribute names, and λ ($\notin \mathbf{att}$) be a special attribute used to denote object identity. Moreover, let $\mathbf{obj}$ be an infinite set of object identifiers (OIDs) and $\mathbf{class}$ be a set of class names. Given a finite set of class names C, a *type* over C is a set $\{A_1 : \tau_1, A_2 : \tau_2, \ldots, A_n : \tau_n\}$, where $A_1, A_2, \ldots, A_n$ are distinct attribute names and each τ_i, with $1 \leq i \leq n$, is either an atomic type or a class name in C. A *schema* is a pair (C, ρ), where C is a finite set of class names and ρ is a total function that maps each class name in C into a type over C. An *OID assignment* to C is a function $\pi : C \rightarrow \wp(\mathbf{obj})$ such that, for every $c, d \in C$, with $c \neq d$, $\pi(c) \cap \pi(d) = \emptyset$. A *tuple* of the type $\{A_1 : \tau_1, \ldots, A_n : \tau_n\}$ over C is a set $\{A_1 : v_1, \ldots, A_n : v_n\}$ such that, for each i, if $\tau_i \in C$, then $v_i \in \pi(\tau_i)$ and

if τ_i is an atomic type, then v_i is a value in its domain. An *instance* of a schema (C, ρ) is a pair $\mathbf{I} = (\pi, \nu)$, where π is an OID assignment to C and ν is a function over $O = \bigcup \{\ \pi(c) \mid\ c \in C\ \}$ that maps each OID in O to a properly-typed tuple. Given an instance $\mathbf{I} = (\pi, \nu)$ of a schema (C, ρ), an *object* of class $c \in C$ is a set $\{\lambda : o, A_1 : v_1, \ldots, A_n : v_n\}$, where $o \in \pi(c)$ and $\nu(o) = \{A_1 : v_1, \ldots, A_n : v_n\}$. For every instance $\mathbf{I}$, there exists a mapping $\overline{\mathbf{I}}$ over C, which associates with each class $c \in C$ the set of objects belonging to it. Objects, tuples, and types can be viewed as total functions over a suitable subset of $\mathbf{att} \cup \{\lambda\}$. The notation $[\![t]\!]$ is used to denote the domain of the function t, that is, an object, a tuple, or a type. $[\![t]\!]_\lambda$ stands for $[\![t]\!] \cup \{\lambda\}$. Moreover, if t is a function and $X \subseteq [\![t]\!]$, then $t[X]$ denotes the total function t' over X such that, for each $x \in X$, $t'(x) = t(x)$. A *temporal instance* is an infinite time series of instances. Formally, a *temporal instance* $\mathbf{T}$ of a schema (C, ρ) is an infinite sequence of instances of (C, ρ). The i^{th} instance of $\mathbf{T}$ is denoted as $\mathbf{T}_i = (\pi_i, \nu_i)$. For any $c \in C$, an element of $\overline{\mathbf{T}}_i(c)$ is an object of c at time i.

The notion of TFD is defined as follows.

Definition 8 *A TFD over the schema (C, ρ) is an expression of the form $c : X \rightarrow_\alpha Y$, where $c \in C$, α is a time relation, and $X, Y \subseteq [\![\rho(c)]\!]_\lambda$, with $X \neq \emptyset$.*

The TFD $c : X \rightarrow_\alpha Y$ is satisfied by a temporal instance $\mathbf{T}$ of (C, ρ) if and only if, for every $(i, j) \in \alpha$, $t_1 \in \overline{\mathbf{T}}_i(c)$, and $t_2 \in \overline{\mathbf{T}}_j(c)$, if $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$.

The meaning of the TFD $c : X \rightarrow_\alpha Y$ can be intuitively explained as follows. Let t_1 and t_2 be two objects of the 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 [11], Wijsen introduces a special notation for some relevant subclasses of TDFs. In particular, he denotes $X \rightarrow_{Next} Y$ by $X\mathbf{N}Y$ and $X \rightarrow_{Forever} Y$ by $X\mathbf{G}Y$.

Wijsen's TFDs allow one to specify conditions on tuples grouped according to any given time relation. As an example, if we consider the schema $(\{Emp\}, \rho)$, where $\rho(Emp) = \{empId : \mathsf{string}, salary : \mathsf{integer}, role : \mathsf{string}\}$, integer and string being atomic types, it is possible to express the condition "*employees cannot have different salaries over two consecutive time points, if their role does not change (but their salaries may change both if they change their role and if they are fired and then re-hired, even with the same role*[1]*)*" by means of the TFD Emp :

[1] When re-hired, the employee gets a different OID.

$empId, role \ \mathbf{N} \ salary$.

Vianu's Dynamic FDs. In [10], 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 one (the "new" instance). State and database sequence are formally defined as follows [10].

Definition 9 *Let $\mathbb{N}$ be the set of natural numbers. A set $S \subseteq \mathbb{N}$ is called* initial *if either $S = \{ i \mid 0 \leq i \leq n \}$ for some $n \in \mathbb{N}$ or $S = \mathbb{N}$. For every initial set S, let $S_0 = S - max(S)$ if $max(S)$ exists, and $S_0 = S$ otherwise.*

Definition 10 *A* database sequence *over a set of attributes U is a sequence $(I_i, \mu_i)_{i \in S}$, where S is an initial set, I_i is an instance over U, for every $i \in S$, μ_i is a partial one-to-one update mapping from I_i to I_{i+1}, for every $i \in S_0$, and $\mu_{max(S)} = \emptyset$, if $max(S)$ exists.*

Tuple are viewed as representations of domain objects. Since a tuple and its updated version represent the same object, tuples must preserve their identity through updates. This is formally achieved by *update mappings*: for each tuple x in I_i, $\mu_i(x)$ is the updated version (if any) of x in I_{i+1}. Moreover, each tuple in I_i which does not belong to the domain of μ_i (dom_{μ_i}) is deleted and each tuple in I_{i+1} which does not belong to the range of μ_i (rng_{μ_i}) is inserted. Properties of the evolution of objects over time are expressed by "dynamic" dependencies, 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 I with its updated version in I'. According to Vianu's notation, for each attribute A, $\overset{\vee}{A}$ represents its old value and $\hat{A}$ its new value. For each set U of attributes, let $\overset{\vee}{U} = \{\overset{\vee}{A} \mid A \in U\}$ and $\hat{U} = \{\hat{A} \mid A \in U\}$. The notion of *action relation* is formally defined as follows.

Definition 11 *The* action relation *associated with an update (I, μ, I') is the relation $I \mu I' = \left\{ \overset{\vee}{x} \times \overset{\wedge}{\mu(x)} \mid x \in I \right\}$.*

Constraints on the evolution of attribute values of tuples (objects) over time are expressed by means of dynamic functional dependencies (DFDs), which are defined as follows.

Definition 12 *A DFD over U is an FD $X \rightarrow Y$ over $\overset{\vee}{U}\hat{U}$ such that, for all $A \in Y$, both $XA \cap \overset{\vee}{U} \neq \emptyset$ and $XA \cap \hat{U} \neq \emptyset$.*

The above condition on DFDs ensures that $X \rightarrow Y$ does not imply any nontrivial FD over $\overset{\vee}{U}$ or $\hat{U}$.

Definition 13 *A DFD $X \rightarrow Y$ is* bipartite *if either $X \subseteq \overset{\vee}{U}$ and $Y \subseteq \hat{U}$, or $X \subseteq \hat{U}$ and $Y \subseteq \overset{\vee}{U}$.*

Informally, a bipartite DFD $\overset{\vee}{X} \rightarrow \hat{Y}$ imposes to tuples that feature the same values for the attributes X at time (state) i to feature the same values for the attributes Y at time (state) $i + 1$.

Definition 14 *An update (I, μ, I') satisfies a set Δ of DFDs, written $(I, \mu, I') \models \Delta$, if the action relation $I \mu I'$ satisfies Δ.*

Definition 15 *Let (U, Σ, Δ) be a DFD schema, where U is a set of attributes, Σ is a set of FDs over U, and Δ is a set of DFDs over U. A database sequence $\{(I_i, \mu_i)_{i \in S}\}$ satisfies (Σ, Δ), denoted $\{(I_i, \mu_i)_{i \in S}\} \models (\Sigma, \Delta)$, if each I_i, with $i \in S$, satisfies Σ and each update $(dom_{\mu_i}, \mu_i, rng_{\mu_i})$, with $i \in S_0$, satisfies Δ.*

Notice that, for any update (I, μ, I') over U, the action relation $I \mu I'$ does not contain distinct tuples that agree on $\overset{\vee}{U}$ or on $\hat{U}$, as tuples are not duplicated in a relation.

As an example, the condition: "*the new salaries of employees uniquely depend on their current and previous roles*" is captured by the DFD $\overset{\vee}{role}\,\overset{\wedge}{role} \rightarrow \overset{\wedge}{salary}$ over the set of attributes $U = \{empId, salary, role\}$.

C Expressive Power

In this section we show how our framework subsumes the TFDs proposed in the literature and described in Section B. Wijsen's TFDs subsumes both Jensen et al.'s and Bettini et al.'s TFDs and it has been proved by Wijsen that the two TFDs proposed by Wijsen [12] and Vianu [10] are orthogonal from the point of view of expressiveness. Then it suffices to prove that our proposal subsumes both Vianu's and Wijsen's TFDs. In order to encode Wijsen's TFDs we have to fill the gap between the two data models since the Wijsen's one is object-based and our is relational-based. Given a schema (C, ρ) in the Wijsen's data model, for every class $c \in C$ we define a temporal schema $R_c^{(C,\rho)}$ with attributes $[\![t]\!]_\lambda \cup \{VT\}$. Given a temporal instance $\mathbf{T} = [(\pi_i, \nu_i)]_{i \in \mathbb{N}}$ over a schema (C, ρ), for every class

$c \in C$, we define the corresponding instance $r_c^{\mathbf{T}}$ of $R_c^{(C,\rho)}$ as the minimal set of tuples which satisfy the following condition:

for every $i \in \mathbb{N}$ and every object $o \in \pi_i(c)$ with $\nu_i(o) = \{A_1 : v_1, \ldots, A_n : v_n\}$ there exists a tuple $t \in r_c^{\mathbf{T}}$ with $t[VT] = i$, $t[\lambda] = o$ and for every $1 \leq j \leq n$ the condition $t[A_j] = v_j$ holds.

Moreover we assume that for every time relation $\alpha \subseteq \mathbb{N} \otimes \mathbb{N}$ there exists a relation R_α consisting of two attributes I and J, both with domain $\mathbb{N}$. For our purposes R_α satisfies the condition $(i, j) \in R_\alpha$ if and only if $(i, j) \in \alpha$ for every $i, j \in \mathbb{N}$. Finally, we can give the following result.

Theorem 1 *For every temporal instance $\mathbf{T} = [(\pi_i, \nu_i)]_{i \in \mathbb{N}}$ over a schema (C, ρ) and every Wijsen's TFD $c : X \rightarrow_\alpha Y$ on the schema (C, ρ) we have that $c : X \rightarrow_\alpha Y$ is satisfied by $\mathbf{T}$ if and only if the TFD $[R_c^{(C,\rho)}, \alpha]X \rightarrow Y$ holds on the instance $r_c^{\mathbf{T}}$ of the temporal schema $R_c^{(C,\rho)}$, where α is a mapping which groups tuples whose valid times are in the α relation.*

Proof Both implications can be proved by contradiction in a very straightforward way. Let us start from the left-to-right implication. Suppose by contradiction that there exist two tuples $\bar{t}_1$ and $\bar{t}_2$ in $R_c^{(C,\rho)}$ such that $\bar{t}_1[X] = \bar{t}_2[X]$, $\bar{t}_1[Y] \neq \bar{t}_2[Y]$ and $(\bar{t}_1[VT], \bar{t}_2[VT]) \in \alpha$. By definition of $r_c^{\mathbf{T}}$ there exist two objects $t_1 = \{\lambda : o_1, A_1 : v_1^1, \ldots, A_n : v_n^1\}$ and $t_2 = \{\lambda : o_2, A_1 : v_1^2, \ldots, A_n : v_n^2\}$ with $o_1 \in \pi_{\bar{t}_1[VT]}(c)$, $o_2 \in \pi_{\bar{t}_2[VT]}(c)$, $\nu_{\bar{t}_1[VT]}(o_1) = \{A_1 : v_1^1, \ldots, A_n : v_n^1\}$ and $\nu_{\bar{t}_2[VT]}(o_2) = \{A_1 : v_1^2, \ldots, A_n : v_n^2\}$. Moreover we have by definition of $r_c^{\mathbf{T}}$ that $t_1[X] = t_2[X]$ and $t_1[Y] \neq t_2[Y]$, this leads to a contradiction. The right-to-left direction is specular. Let $t_1 = \{\lambda : o_1, A_1 : v_1^1, \ldots, A_n : v_n^1\}$ and $t_2 = \{\lambda : o_2, A_1 : v_1^2, \ldots, A_n : v_n^2\}$ two objects for which there exists $(i, j) \in \alpha$ with $o_1 \in \pi_i(c)$, $o_2 \in \pi_j(c)$, $\nu_i(o_1) = \{A_1 : v_1^1, \ldots, A_n : v_n^1\}$ and $\nu_j(o_2) = \{A_1 : v_1^2, \ldots, A_n : v_n^2\}$. By definition of $r_c^{\mathbf{T}}$ there exist two tuples $\bar{t}_1$ and $\bar{t}_2$ in $r_c^{\mathbf{T}}$ with $\bar{t}_1[\lambda] = o_1$, $\bar{t}_2[\lambda] = o_2$, $\bar{t}_1[A_j] = v_j^1$ and $\bar{t}_2[A_j] = v_j^2$ for $1 \leq j \leq n$, $\bar{t}_1[VT] = i$ and $\bar{t}_2[VT] = j$. Then we have $\bar{t}_1[X] = \bar{t}_2[X]$ and $\bar{t}_1[Y] \neq \bar{t}_2[Y]$ (contradiction).

Now we concentrate ourselves on the Vianu's DFDs. In [10] the author adopt a relational-based data-model enriched with an update mapping function μ_i as described in Section B; Vianu, keeping an higher level of abstraction, does not mention how this function is implemented in a real database. We may think that this function is embedded directly into the relational schema R by adding an additional attribute TID to the set of atemporal attributes U. The TID attribute is a *snapshot key* for the relation R and it means that given any istance r of R we have that for every $i \in \mathbb{N}$ the standard functional dependency $TID \rightarrow U$ holds over the snapshot $\{t \mid r(t) \wedge t[VT] = i\}$. Given a database sequence $\{(I_i, \mu_i)_{i \in S}\}$ for

some initial set S, we define the corresponding instance r of a relation schema R with atemporal attributes $U \cup \{TID\}$ as the smallest set of tuples satisfying the following conditions:

1. for every $i \in S$ and every $t \in I_i$ there exists a tuple $\bar{t} \in r$ with $\bar{t}[U] = t[U]$, $\bar{t}[VT] = i$;

2. for every $i \in S$ the standard functional dependency $TID \rightarrow U$ holds over the snapshot $\{t \mid r(t) \wedge t[VT] = i\}$;

3. for every $i \in S_0$ and for every tuple $t \in dom_{\mu_i}$ we have that there exists two tuples $\bar{t}, \bar{t}'$ such that $\bar{t}[U] = t[U], \bar{t}'[U] = \mu_i(t)[U], \bar{t}[VT] = i, \bar{t}'[VT] = i{+}1$ and $\bar{t}[TID] = \bar{t}'[TID]$;

4. for every $i \in S_0$ and for every tuple $t \in I_i \setminus dom_{\mu_i}$ we have that the instance $\{\bar{t} \mid r(\bar{t}) \wedge \bar{t}[VT] = i+1 \wedge \exists \bar{t}'(r(\bar{t}') \wedge \bar{t}'[U] = t[U] \wedge \bar{t}'[VT] = i \wedge \bar{t}'[TID] = \bar{t}[TID])\}$ is empty.

Now we are ready to give the following result.

Theorem 2 *Let (U, Σ, Δ) be a DFD schema, a database sequence $\{(I_i, \mu_i)_{i \in S}\}$ satisfies $\overset{\vee}{X}\hat{W} \rightarrow \overset{\vee}{Z}\hat{Y}$ if and only if the corresponding instance r of the relational schema R with attributes $U \cup \{TID, VT\}$ satisfies $[\chi^R_{TID}, \mathsf{Top(i)}]X\overline{W} \rightarrow Z\overline{Y}$.*

Proof The proof is by contradiction for both implications. For the left-to-right implication we assume by contradiction that the corresponding instance r does not satisfy the TFD $[\chi^R_{TID}, \mathsf{Top(i)}]X\overline{W} \rightarrow Z\overline{Y}$ which means that there exists two tuples in $\bar{t}_1$ and $\bar{t}_2$ in the instance r^χ of χ^R_{TID} with $\bar{t}_1[X\overline{W}] = \bar{t}_2[X\overline{W}]$ and $\bar{t}_1[Z\overline{Y}] \neq \bar{t}_2[Z\overline{Y}]$. By construction of χ^R_{TID} we have that there exists four tuples two natural numbers i and j and four tuples $\overline{s}^i_1, \overline{s}^{i+1}_1, \overline{s}^j_2$ and $\overline{s}^{j+1}_2$ with $\overline{s}^i_1[VT] = i$, $\overline{s}^{i+1}_1[VT] = i+1$, $\overline{s}^j_2[VT] = j$, $\overline{s}^{j+1}_2[VT] = j+1$, $\overline{s}^i_1[TID] = \overline{s}^{i+1}_1[TID]$, $\overline{s}^j_2[TID] = \overline{s}^{j+1}_2[TID]$, $\overline{s}^i_1[X] = \overline{s}^j_2[X] = \bar{t}_1[X]$, $\overline{s}^{i+1}_1[W] = \overline{s}^{j+1}_2[W] = \bar{t}_1[\overline{W}]$ and either $\overline{s}^i_1[Z] \neq \overline{s}^j_2[Z]$ or $\overline{s}^{i+1}_1[Y] \neq \overline{s}^{j+1}_2[Y]$ holds. By definition of r we have that there exist (I_i, μ_i) and (I_j, μ_j) and two tuples $t_1 \in I_i$ and $t_2 \in I_j$ with $t_1 \in dom_{\mu_i}$, $t_2 \in dom_{\mu_j}$, $t_1[X] = t_2[X] = \bar{t}_1[X]$, $\mu_i(t_1)[W] = \mu_j(t_2)[W] = \bar{t}_1[W]$ and either $t_1[Z] \neq t_2[Z]$ or $\mu_i(t_1)[Y] \neq \mu_j(t_2)[Y]$, this leads to a contradiction. For the right-to-left implication suppose that there exists two natural numbers $i, j \in \mathbb{N}$ and two tuples $t_1 \in I_i$ and $t_2 \in I_j$ with $t_1 \in dom_{\mu_i}$, $t_2 \in dom_{\mu_j}$, $t_1[X] = \bar{t}_2[X] = t_1[X]$, $\mu_i(t_1)[W] = \mu_j(t_2)[W] = t_1[W]$ and either $t_1[Z] \neq t_2[Z]$ or $\mu_i(t_1)[Y] \neq \mu_j(t_2)[Y]$. By definition of r, it contains four tuples $\overline{s}^i_1, \overline{s}^{i+1}_1, \overline{s}^j_2$ and $\overline{s}^{j+1}_2$ with $\overline{s}^i_1[VT] = i$, $\overline{s}^{i+1}_1[VT] = i+1$, $\overline{s}^j_2[VT] = j$, $\overline{s}^{j+1}_2[VT] = j+1$, $\overline{s}^i_1[TID] = \overline{s}^{i+1}_1[TID]$, $\overline{s}^j_2[TID] = \overline{s}^{j+1}_2[TID]$, $\overline{s}^i_1[X] = \overline{s}^j_2[X] = t_1[X]$, $\overline{s}^{i+1}_1[W] = \overline{s}^{j+1}_2[W] = \mu_i(t_1)[W]$ and either $\overline{s}^i_1[Z] \neq \overline{s}^j_2[Z]$ or $\overline{s}^{i+1}_1[Y] \neq \overline{s}^{j+1}_2[Y]$

holds. By definition of χ^R_{TID} in its instance r^χ there exists two tuples $\bar{t}_1$ and $\bar{t}_2$ with $\bar{t}_1[X] = \bar{t}_2[X] = t_1[X]$ and $\bar{t}_1[\overline{W}] = \bar{t}_2[\overline{W}] = \mu_i(t_1)[W]$ and either $\bar{t}_1[Z] \neq \bar{t}_2[Z]$ or $\bar{t}_1[\overline{Y}] \neq \bar{t}_2[\overline{Y}]$ and this leads to a contradiction.

University of Verona
Department of Computer Science
Strada Le Grazie, 15
I-37134 Verona
Italy

http://www.di.univr.it