A uniform algebraic characterization of temporal functional dependencies*

C. Combi$^+$, A. Montanari*, R. Rossato$^+$
$^+$Dipartimento di Informatica, Università degli Studi di Verona, Italy
{*combi,rossato*}@*sci.univr.it*
* Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Italy
montana@*dimi.uniud.it*

Abstract

In the database literature, different types of temporal functional dependencies (TFDs) have been proposed to constrain the temporal evolution of information. Unfortunately, the lack of a common notation makes it difficult to compare, to integrate, and to possibly extend the various proposals. In this paper, we outline a unifying algebraic framework for TFDs. We first introduce the proposed approach, then we use it to give a uniform account of existing TFDs, and finally we show that it allows one to easily express new meaningful TFDs.

1 Introduction

Data dependencies provide a semantic specification of the meaningful instances of a given database/relation scheme. *Functional dependencies* (FDs) are a specific class of data dependencies which play an important role in the logical design of relational databases [12]. Formally, given a relation scheme $R(Z)$, a functional dependency FD over $R(Z)$ is an expression of the form $X \to Y$, with $X, Y \subseteq Z$, which is satisfied by a relation $r \in R(Z)$ if and only if for every $t, t' \in r$, if $t[X] = t'[X]$, then $t[Y] = t'[Y]$. FDs have been developed in the context of databases devoid of any temporal dimension of data, and thus they provide no direct support to the management of temporal constraints. Temporal databases allow one to describe the temporal evolution of information by associating one or more temporal dimensions with the stored data [7]. *Temporal functional dependencies* (TFDs) have been used in temporal databases to constrain the temporal relationships among data [1, 10, 11, 14, 15, 16, 17]. A few of them make it possible to express temporal constraints at different levels of time granularity [1, 16]. Some TFDs have been used during database design to generalize the conventional normal-form-based decomposition of relational schemes to the temporal setting, others have been used to express database (temporal) integrity constraints. In the following, we will focus our attention on this latter use. Unfortunately, the lack of a common notation makes it difficult to compare, to integrate, and to possibly extend the various proposals.

In this paper, we outline an algebraic approach that allows us to give a uniform account of existing TFDs. It is well-known that (atemporal) FDs can be turned into containment equations whose satisfaction can be checked by testing suitable relational algebra expressions for emptiness [12]. In the following, we show that the simplest TFDs can be directly expressed in terms of atemporal FDs. This is not the case, however, with the most complex TFDs. To cope with them, we need an auxiliary procedure which generates a set of atemporal relations (the so-called snapshot relations, snapshots for short) on which atemporal FDs must be checked. We show that a uniform solution can be obtained by reducing the satisfaction problem for TFDs to the problem of establishing whether the evaluation of suitable relational algebra expressions over the *temporal database* returns the empty set. As a matter of fact, the most difficult cases are those of TFDs involving different time granularities.

The paper is organized as follows. In Section 2, we briefly survey the different classes of existing TFDs. In Section 3, we introduce a clinical application to be used to exemplify the proposed solution. Next, in Section 4 we outline our algebraic approach, and we apply it to the various classes of TFDs. In Section 5, we show that new meaningful TFDs can be easily expressed in the algebraic framework. Section 6 summarizes the achieved results, and it briefly discusses further research directions.

*This work has been partially supported by contributions from the Italian Ministry of University and Research (MIUR) under the programs COFIN-PRIN “Constraint-based verification of reactive systems” and “Intelligent analysis of hemodialysis data for the improvement of care processes”.

2 An overview of existing TFDs

TFDs devoid of time granularities. In [14] Vianu represents a temporal relation as a temporal sequence of snapshots. A new snapshot is obtained from the current one by a global update, which affects some, possibly all, its tuples simultaneously. According to Vianu's view, each snapshot can be interpreted as a set of objects, each one denoted by a distinct tuple, which preserve their identity through updates. He distinguishes two classes of data dependencies, respectively called *static* and *dynamic* dependencies. Static dependencies are standard (atemporal) FDs that must be satisfied by any single snapshot, while dynamic dependencies express conditions on the evolution of the database, that is, on the sequence of snapshots.

In [11] Jensen et al. propose a bitemporal data model that associates a *bitemporal element* (v, t) with each tuple, where v and t are the values of the two basic temporal dimensions of *valid time* and *transaction time*, respectively [7]. The valid time of a fact is the time when the fact is true in the modeled reality, while the transaction time is the time when the fact is current in the database and it can be retrieved. The scheme of a bitemporal relation is obtained by adding a special temporal attribute to the set of traditional attributes which temporally qualifies a tuple by means of a bitemporal element. In fact, we can view a bitemporal element as a set of pairs of values associated with the valid and transaction time attributes (and thus bitemporal relations can be expressed in 1NF). FDs are extended to temporal relations as follows. Given a bitemporal relation scheme R(Z) and two sets of non-temporal attributes $X, Y \subseteq Z$, a relation $r \in R(Z)$ satisfies a *temporal functional dependency* $X \rightarrow_T Y$ if all snapshots of *r* satisfy the atemporal FD $X \rightarrow Y$ (it is thus similar to *static dependency* introduced by Vianu [14]). In a subsequent paper [10], Jensen and Snodgrass introduce a set of notions to be used to express temporal aspects of a scheme, and they show how to use them in database design. In particular, they define suitable notions of temporal key and temporal normal form.

Finally, in [15] Wijsen proposes a data model extended with valid time, where time is represented as a discrete and bounded set of time points **TIME** = $\{0, 1, 2, \ldots,$ *maxTime*$\}$. The valid time of a tuple is expressed by means of a special attribute VT whose value is the set of time points at which the tuple is true in the reality. Given a valid time relation r, a *snapshot* r_i of r at time *i*, with $i \in \mathbb{N}$, is the set of valid tuples of r at time i, devoid of their temporal component. A *snapshot functional dependency* (FD) $X \rightarrow Y$ is satisfied by a valid time relation r at time *i* if and only if it is satisfied by r_i. Keys and normal forms are defined in term of FDs in the standard way. Besides FDs, Wijsen introduces two classes of temporal dependencies. *Temporal functional dependencies* (TFDs) are of the form *X **G** Y*, where X and Y are sets of attributes. A TFD *X **G** Y* is satisfied by a valid time relation r at time *i* if and only if the FD $X \rightarrow Y$ is satisfied by $r_i \cup r_{i+1}, \ldots, r_{maxTime}$. *Dynamic functional dependencies* (DFDs) are of the form *X **N** Y*, where X and Y are sets of attributes. A DFD *X **N** Y* is satisfied by a valid time relation r at time *i*, if and only if the FD $X \rightarrow Y$ is satisfied by $r_i \cup r_{i+1}$ (by $r_{maxTime}$ if *i* = *maxTime*).

TFDs involving multiple time granularities. A time granularity partitions the temporal domain T in various groups of indivisible elements, called *granules*, where each group corresponds to a distinct time unit. In temporal databases, granularity systems have been exploited to specify temporal constraints about differently-grained elements of standard calendars, such as the Gregorian one, as well as elements of user-defined calendars, that is, finite sets of time granularities. A *time granularity* G is a function from an index set I_G to the powerset of the temporal domain $\wp(T)$ such that for any pair of granules $G(i)$ and $G(j)$, with $i \neq j$, (i) $G(i) \cap G(j) = \emptyset$ and (ii) if $i < j$, then for all $t \in G(i)$ and $t' \in G(j)$ $t < t'$. Moreover, the subset of I_G that maps indices to nonempty granules forms an initial segment of I_G. Various relations can be defined between pairs of granularities, such as, for instance, grouping, refinement, and partition [1].

In [1] Bettini et al. define the notion of *temporal functional dependency* (TFD) *with multiple time granularities*. A time granularity is defined as a mapping μ from the index set $\mathbb{N}$, whose elements are called time points, to the powerset $\wp(\mathbb{R})$ of the absolute temporal domain $\mathbb{R}$. A *temporal module scheme* is a pair (R, G), where R is a relation scheme and G is a time granularity. A *temporal module* is a triple (R, G, ϕ), where (R, G) is a temporal module scheme and ϕ is a function, called a *time windowing function*, that associates a set of tuples with any granule $G(i)$/time point i (the set of tuples valid at i). TFDs are of the form $X \rightarrow_G Y$, and they properly extend atemporal FDs. By definition, a temporal module (R, G, ϕ) satisfies a TFD $X \rightarrow_{G'} Y$, with G' coarser than G, if and only if whenever two tuples, that hold on time granules in G covered by the same granule in G', agree on X, they must also agree on Y. On the basis of such TFDs, they propose a temporal normalization theory together with suitable algorithms for normal-form-based decomposition.

In [16] Wijsen introduces a notion of *temporal functional dependency* (TFD) for temporal databases extended with time granularity which is quite similar to Bettini et al.'s one [1]. The distinctive concepts of the underlying temporal model are those of *temporal relation* and *object identity*. The temporal domain is assumed to be (isomorphic to) $\mathbb{N}$ and its elements are called time points. A time granularity is defined as a suitable binary relation on $\mathbb{N}$, called temporal relation. Furthermore, the relational model is extended 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 (this is obtained by making it possible to associate with each object attribute either an atomic value or a pointer to another object).

3 A clinical application

In this section we briefly introduce a real-world example taken from clinical medicine, namely, chemotherapies for oncological patients, that we shall later use to exemplify the solution we propose. As a general rule, oncological patients undergo several chemotherapy cycles, and each cycle typically includes the administration of several drugs. The temporal schedule of drug administrations within each cycle is usually predefined and the same cycle can be repeated several times. Let us consider, for instance, the following chemotherapy recommendation for Hoadkin's disease [9]:

"The ChlVPP regimen consists of chlorambucil (6 mg/m^2/day) on days 1 through 14, vinblastine (6 mg/m^2) on days 1 and 8, procarbazine (100 mg/m^2/day) on days 1 through 14, and prednisone (40 mg/day) on days 1 through 14. Patients treated with this regimen receive six cycles repeated every 28 days."

This therapy plan implicitly introduces some time granularities, which define the time scheduling for the assumption of the prescribed drugs. They are graphically depicted in Figure 1, where we use the shorthands ChlVPP, Chl, V, Pc, and Pd for ChlVPP regimen, chlorambucil, vinblastine, procarbazine, and prednisone, respectively.

Let $T\_Patient$ be a temporal relation that stores information about the patients who underwent ChlVPP chemotherapies, and let $Patient$ be its atemporal component with scheme $Patient(P\text{-}Id, Chemo, BG, Drug, Qty)$, where *P-Id* is the patient identifier, *Chemo* is the type of therapy, *BG* is the blood group of the patient, and *Drug* and *Qty* are respectively the name of the assumed drug and its quantity. Let the valid time VT of each tuple be the time of the specific drug assumption, expressed as the number of days from the beginning of the therapy. The temporal relation $T\_Patient$ pairs information about patients with the time of their drug assumption. The scheme of $T\_Patient$ is obtained by adding the temporal attribute *VT* to the set of attributes of the scheme $Patient$, that is, $\mathsf{att}(T\_Patient) = \mathsf{att}(Patient) \cup \{VT\}$.

4 An algebraic approach for TFDs

In this section, we first show how TFDs can be brought back to (atemporal) FDs over atemporal relations, obtained by means of suitable algebraic operations. As a matter of fact, for some TFDs it is necessary to provide an external procedure that generates all the required relations. Next, we show that the problem of checking whether a temporal database satisfies a given (set of) TFD(s) can be always reduced to the problem of establishing whether the evaluation of a suitable relational algebra query over the temporal database returns the empty relation.

P-Id	***Chemo***	***BG***	***Drug***	***Qty***	***VT***
p_1	ChlVPP	0+	Chlo	6	1
p_1	ChlVPP	0+	Vinb	6	1
p_1	ChlVPP	0+	Proc	100	1
p_1	ChlVPP	0+	Pred	40	1
p_1	ChlVPP	0+	Chlo	6	2
p_1	ChlVPP	0+	Proc	100	2
p_1	ChlVPP	0+	Pred	40	2
...	...	...	...	...	...
p_1	ChlVPP	0+	Chlo	6	8
p_1	ChlVPP	0+	Vinb	6	8
p_1	ChlVPP	0+	Proc	100	8
p_1	ChlVPP	0+	Pred	40	8
...	...	...	...	...	...

Table 1. A $T\_Patient$ instance that records data about the first cycle of treatment for p_1.

To clarify the proposed solution, we will make use of some examples of TFDs over $T\_Patient$. Table 1 shows a $T\_Patient$ instance that stores data about the first cycle of chemotherapy for patient (with identifier) p_1. The attribute VT encodes the valid time in term of days from the beginning of the therapy. As an example, the first four tuples store information about the assumption of the drugs Chlo, Vinb, Proc, and Pred on the first day of the cycle (VT=1). They can be retrieved by executing the following query:

$$\pi_{P\text{-}Id,Chemo,BG,Drug,Qty}(\sigma_{VT=1}(T\_Patient))$$

In general, the *snapshot* r_i, at time i, contains all and only the tuples of a temporal relation T_r valid at time i over its atemporal attributes Z, that is, $r_i = \pi_Z(\sigma_{VT=i}(T\_r))$.

4.1 Vianu's dynamic dependencies

In [14] Vianu proposes a simple extension to the relational model aimed at capturing the temporal evolution of a database. It views a temporal database as a temporal sequence of database instances (states). Updates, insertions, and deletions force the transition from one state to the next one. More precisely, a temporal database is defined as a sequence of consecutive database instances and a sequence of "update mappings" that link pairs of consecutive instances

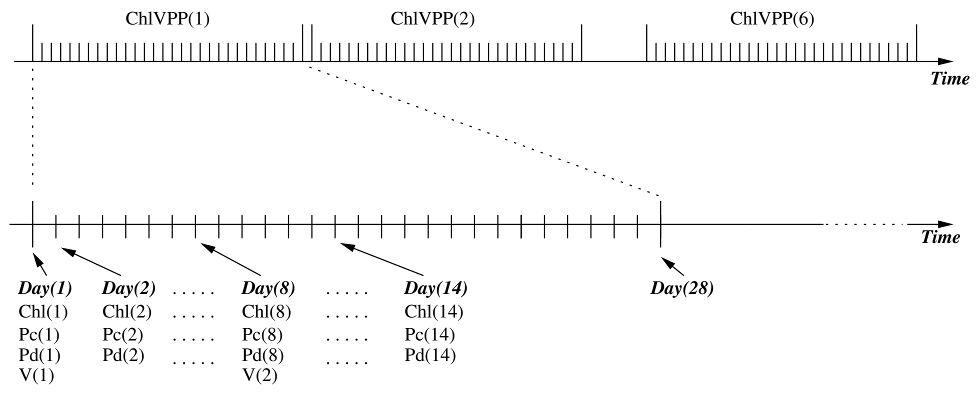

Figure 1. Granularities involved in the chemotherapy treatment.

(old/new instances). A tuple is viewed as (a representation of) an object. Since a tuple and its updated version denote the same object, every tuple preserves its identity through updates. This is guaranteed by the *update mappings* μ_i: for every tuple x belonging to the instance I_i, $\mu_i(x)$ is the updated version of x belonging to I_{i+1}. Conditions on the temporal evolution of objects are expressed by means of *dynamic dependencies*.

According to Vianu's notation, we will denote the old and the new value of an attribute A by $\overset{\vee}{A}$ and $\hat{A}$, respectively. Furthermore, to preserve the identity of tuples through updates, Vianu extends the scheme of a relation by adding a special attribute λ (a placeholder for object identifiers), whose value allows one to link a tuple and its updated version. Given a temporal relation T_r, we denote by T_r^* the relation obtained by adding the attribute λ to the scheme of T_r (cf. Table 2):

$$\mathsf{att}(T\_r^*) = \{\lambda\} \cup \mathsf{att}(T\_r)$$

Dynamic dependencies relate every relation instance (snapshot) to its successor. More precisely, a dynamic dependency $\overset{\vee}{X} \to \hat{Y}$ imposes to tuples that feature the same values for the attributes X at time i to feature the same values for the attributes Y at time $i + 1$.

We now show that the problem of checking whether a dynamic dependency is satisfied by a given temporal relation can be reduced to that of checking whether a functional dependency over a suitable atemporal relation is satisfied. Such an atemporal relation is obtained by joining the snapshot at time i with the snapshot at time $i + 1$, once the attributes of the latter have been properly renamed. Let r_i be the snapshot of T_r^* at time i returned by the query:

$$\pi_{\{\lambda\} \cup Z}(\sigma_{VT=i}(T\_r^*))$$

and let $\overset{\sim}{r_{i+1}}$ be the snapshot of T_r^* at time $i+1$, where the attributes $a_1, \ldots, a_k$ have been replaced by the attributes $b_1, \ldots, b_k$:

$$\rho_{b_1 \ldots b_k | a_1 \ldots a_k}(r_{i+1})$$

By a natural join on λ, we can merge the two snapshot relations r_i and $\overset{\sim}{r_{i+1}}$ into an atemporal relation $r_{V_{i,i+1}}$, whose tuples collect the values of the original temporal relation immediately before and immediately after the update:

$$r_{V_{i,i+1}} = r_i \bowtie \overset{\sim}{r_{i+1}}$$

Let $\tilde{Y}$ be the result of the renaming of the attributes in Y. We have that the dynamic dependency $\overset{\vee}{X} \to \hat{Y}$ is satisfied by the temporal relation T_r^* if and only if the functional dependency $X \to \tilde{Y}$ is satisfied by the relations $r_{V_{i,i+1}}$ for every time point i. Note that we need an external procedure to generate, for every time point i, the relation $r_{V_{i,i+1}}$ to test. Such a procedure is not necessary anymore if we directly work on the given temporal relation. The dynamic dependency $\overset{\vee}{X} \to \hat{Y}$ can indeed be encoded by means of the following relational algebra query. Hereafter, for $i \geq 1$, let $T\_r_i^*$ be the temporal relation obtained from T_r^* by replacing every attribute A (resp. set of attributes X) of T_r^* by A_i (resp. X_i). We have that $\overset{\vee}{X} \to \hat{Y}$, with $X \cap Y = \emptyset$, is satisfied by T_r^* if and only if the query

$$\sigma_{Y_1 \neq Y_3}(Q \bowtie_{(X=X_2 \wedge VT=VT_2)} S),$$

where[1]

$$\begin{array}{lcl} Q & \equiv & T\_r^* \bowtie_{(\lambda=\lambda_1 \wedge VT+1=VT_1)} T\_r_1^* \\ S & \equiv & T\_r_2^* \bowtie_{(\lambda_2=\lambda_3 \wedge VT_2+1=VT_3)} T\_r_3^*, \end{array}$$

returns the empty set.

[1]To simplify the structure of queries, we use the shorthand $+1$.

The join operation merges tuples which feature the same values for the attributes X and X_1, which concern the evolution of the same object, that is, have the same values for λ and λ_1, and are valid on two successive time points. More complex join operations are needed when we need to consider the same attribute(s) before and after the update, as shown by the following example.

Example. Let us extend the scheme of the temporal relation $T\_Patient$ with the special attribute λ. Let $Patient^*$ be the resulting temporal relation. We have that $\mathsf{att}(T\_Patient^*) = \mathsf{att}(Patient^*) \cup \{VT\}$, where the atemporal scheme of $Patient^*$ is:

$$Patient^*(\lambda, P\_Id, Chemo, BG, Drug, Qty).$$

A $T\_Patient^*$ instance is reported in Table 2. Its first tuple (*oid1, p1, ChlVPP, 0+, Chlo, 6, 1*) states that the patient (with identifier) $p1$ assumed the drug chlorambucil on the first day. The special attribute λ plays the role of tuple-identifier. The values $oid1$, $oid2$, $oid3$, and $oid4$ for the attribute λ identify the tuples associated with the assumption of the drugs Chlo, Vinb, Proc, and Pred, respectively.

λ	***P-Id***	***Chemo***	***BG***	***Drug***	***Qty***	***VT***
oid1	p_1	ChlVPP	0+	Chlo	6	1
oid2	p_1	ChlVPP	0+	Vinb	6	1
oid3	p_1	ChlVPP	0+	Proc	100	1
oid4	p_1	ChlVPP	0+	Pred	40	1
oid1	p_1	ChlVPP	0+	Chlo	6	2
oid3	p_1	ChlVPP	0+	Proc	100	2
oid4	p_1	ChlVPP	0+	Pred	40	2
...	...	...	...	...	...	...
oid1	p_1	ChlVPP	0+	Chlo	6	8
oid2	p_1	ChlVPP	0+	Vinb	6	8
oid3	p_1	ChlVPP	0+	Proc	100	8
oid4	p_1	ChlVPP	0+	Pred	40	8
...	...	...	...	...	...	...

Table 2. A $T\_Patient^*$ instance that stores data about the first cycle of treatment for p_1.

Consider the following condition on the relation $T\_Patient^*$: *"each new drug is determined uniquely on the basis of the current type of chemotherapy and the assumed drug"*. According to Vianu's notation, this condition can be expressed by the temporal dependency: $\overset{\vee}{Chemo}, \overset{\vee}{Drug} \longrightarrow \overset{\wedge}{Drug}$. We have that the temporal relation $T\_Patient^*$ satisfies the given dynamic dependency if and only if, for every i, the atemporal dependency $Chemo, Drug \rightarrow NDrug$ is satisfied by the atemporal relation:

$$r_{V_{i,i+1}} = r_i \bowtie \overset{\sim}{r_{i+1}}$$

where

$$\begin{aligned} r_i &= \pi_{\{\lambda\}\cup Z}(\sigma_{VT=i}(T\_Patient^*)) \\ \overset{\sim}{r_{i+1}} &= \rho_{Cond}(r_{i+1}) \end{aligned}$$

and

$$\begin{aligned} Cond \equiv\ & NP\_Id, NChemo, NBG, NDrug, NQty| \\ & P\_Id, Chemo, BG, Drug, Qty \end{aligned}$$

As already pointed out, we need an external procedure to generate, for $i = 1, \ldots, 28$, the relation $r_{V_{i,i+1}}$ and to check the satisfaction of the given functional dependency on it. As an alternative, the problem of checking the satisfaction of $\overset{\vee}{Chemo}, \overset{\vee}{Drug} \longrightarrow \overset{\wedge}{Drug}$ can be brought back to the problem of establishing whether the query

$$\sigma_{Drug_1 \neq Drug_3}(Q \bowtie_{Cond} S),$$

where

$$\begin{aligned} Q &\leftarrow T\_Patient^* \bowtie_{(\lambda=\lambda_1 \wedge VT+1=VT_1)} T\_Patient_1^* \\ S &\leftarrow T\_Patient_2^* \bowtie_{(\lambda_2=\lambda_3 \wedge VT_2+1=VT_3)} T\_Patient_3^* \\ Cond &\equiv Chemo = Chemo_2 \wedge Drug = Drug_2 \wedge \\ & \quad VT = VT_2, \end{aligned}$$

returns the empty set.
Note that, in this case, the query must take into account the fact that the intersection of the attributes in the premise and those in the consequence of the dependency is not empty.

4.2 Jensen, Snodgrass, and Soo's TFDs

In [11] Jensen, Snodgrass, and Soo address the problem of expressing (temporal) functional dependencies on data provided with both valid and transaction time dimensions. A bitemporal instance of the relation *Patient* that stores data about the first cycle of the therapy is given in Table 3. We assume the transaction time of the tuples to be equal to *now* to indicate that they are current in the database. Furthermore, we assume a point-based point of view according to which the bitemporal elements (t, v), associated with each tuple, are represented by means of two different temporal attributes (TT and VT, respectively).

Jensen et al. define the satisfaction of TFDs in terms of the satisfaction of the corresponding atemporal FDs: a TFD $X \rightarrow_T Y$ is satisfied by a bitemporal relation r^B if and only if every atemporal instance of r^B satisfies the FD $X \rightarrow Y$. The point-based representation of a bitemporal relation T_r^B makes it possible to reduce the satisfaction of $X \rightarrow_T Y$ over T_r^B to the satisfaction of $X \rightarrow Y$ over the following relations $r_{i,j}$:

$$r_{i,j} = \pi_Z(\sigma_{(TT=j \wedge VT=i)}(T\_r^B))$$

P-Id	*Chemo*	*BG*	*Drug*	*Qty*	*T*
p_1	ChlVPP	0+	Chlo	6	{(*now*,1),(*now*,2),(*now*,3), ..., (*now*,14)}
p_1	ChlVPP	0+	Vinb	6	{(*now*,1),(*now*,8)}
p_1	ChlVPP	0+	Proc	100	{(*now*,1),(*now*,2),(*now*,3), ..., (*now*,14)}
p_1	ChlVPP	0+	Pred	40	{(*now*,1),(*now*,2),(*now*,3), ..., (*now*,14)}

Table 3. A *T_Patient* bitemporal instance.

For all i, j, the relation $r_{i,j}$ consists of the set of tuples current at time j ($TT = j$) which are valid at time i ($VT = i$). Thus, to check the satisfaction of the TFD $X \rightarrow_T Y$ on T_r^B, we need to generate, for all i, j, the relation $r_{i,j}$ and to check whether it satisfies the FD $X \rightarrow Y$.

The use of such an auxiliary procedure can be avoided as follows. We can reduce the problem of checking the given TFD to the equivalent problem of establishing whether the following query returns the empty set:

$$\sigma_{Y \neq Y_1}(T\_r^B \bowtie_{(X=X_1 \,\wedge\, VT=VT_1 \,\wedge\, TT=TT_1)} T\_r_1{}^B)$$

The join operation merges the tuples which are valid at the same time and current at the same time, which feature the same values for the attributes X and X_1.

Example. The TFDs proposed by Jensen et al. can be used to express the following constraint: *"at one time, a patient can undergo only one chemotherapy"* ($P\text{-}Id \rightarrow_T Chemo$). The FD $P\text{-}Id \rightarrow Chemo$ must be checked on all the snapshots of the relation $T\_Patient^B$, that is, for each valid time point i and each transaction time point j of a tuple, we must show that the relation

$$r_{i,j} = \pi_Z(\sigma_{TT=j \,\wedge\, VT=i}(T\_Patient^B))$$

satisfies the FD. Once more, the problem of checking the satisfaction of $P\text{-}Id \rightarrow Chemo$ on $T\_Patient^B$ can be alternatively brought back to the problem of checking whether the following query returns the empty set:

$$\sigma_{Chemo \neq Chemo_1}(T\_Patient^B \bowtie_{Cond} T\_Patient_1^B),$$

where

$$Cond \equiv (P\text{-}Id = P\text{-}Id_1 \;\wedge\; VT = VT_1 \;\wedge\; TT = TT_1)$$

4.3 Wijsen's TFDs

The satisfaction of Wijsen's *temporal functional dependencies* and *dynamic functional dependencies* are expressed in terms of satisfaction of functional dependencies on a specific set of snapshots [15]. According to Wijsen's definition [15], a *temporal functional dependency* $X\mathbf{G}Y$ is satisfied by a temporal relation T_r at time i if the functional dependency $X \rightarrow Y$ is satisfied by $r_i \cup r_{i+1} \ldots \cup r_{maxTime}$. It is satisfied by a temporal relation T_r if and only if it is satisfied by T_r at every time $j \in \mathbf{TIME}$.

We reduce the satisfaction of a TFD $X\mathbf{G}Y$ over the relation T_r at time i to the satisfaction of the FD $X \rightarrow Y$ over the following relation $r_{\geq i}$:

$$r_{\geq i} = \pi_Z(\sigma_{VT \geq i}(T\_r))$$

The relation $r_{\geq i}$ contains the valid tuples of T_r from time i to time *maxTime*. To check the satisfaction of a TFD $X\mathbf{G}Y$ over a temporal relation T_r, we have to check the satisfaction of $X\mathbf{G}Y$ at each time $j \in \mathbf{TIME}$. To this end, it suffices to check the satisfaction of the FD $X \rightarrow Y$ on the relation $r_{\geq 1} = \pi_Z(\sigma_{VT \geq 1}(T\_r)) = \pi_Z(T\_r)$ (according to Table 1, the first time point is numbered by 1).

Equivalently, we can reduce the problem of checking the satisfaction of a TFD $X\mathbf{G}Y$ over the temporal relation T_r to the problem of establishing whether the following query returns the empty set:

$$\sigma_{Y \neq Y_1}(T\_r \bowtie_{X=X_1} T\_r_1)$$

Example. Let us consider the following condition on the relation $T\_Patient$: *"every patient has a value of the blood group which remains unchanged all over the time"*. This dependency can be expressed in Wijsen's notation as $\{P\text{-}Id\}\mathbf{G}\{BG\}$. The relation $T\_Patient$ satisfies $\{P\text{-}Id\}\mathbf{G}\{BG\}$ if (and only if) the dependency $P\text{-}Id \rightarrow BG$ is satisfied by

$$r_{\geq 1} = \pi_Z(\sigma_{VT \geq 1}(T\_Patient)) = \pi_Z(T\_Patient)$$

Alternatively, checking the satisfaction of $\{P\text{-}Id\}\mathbf{G}\{BG\}$ over $T\_Patient$ can be brought back to checking whether the following query returns the empty set:

$$\sigma_{BG \neq BG_1}(T\_Patient \bowtie_{P\text{-}Id=P\text{-}Id_1} T\_Patient_1)$$

According to Wijsen's definition [15], the *dynamic functional dependency* $X\mathbf{N}Y$ is satisfied at time i if the functional dependency $X \rightarrow Y$ is satisfied by $r_i \cup r_{i+1}$. A dynamic functional dependency is satisfied by a temporal relation T_r if and only if it is satisfied by T_r at every time $j \in \mathbf{TIME}$.

We reduce the satisfaction of $X\mathbf{N}Y$ over the temporal relation T_r at time i, with $i < maxTime$, to the satisfaction of $X \rightarrow Y$ over the following relation $r_{i,i+1}$:

$$r_{i,i+1} = \pi_Z(\sigma_{VT=i \vee VT=i+1}(T\_r))$$

If $i = maxTime$, the dependency has to be checked on the relation $\pi_Z(\sigma_{VT=maxTime}(T))$. To check the satisfaction of $X\mathbf{N}Y$ on T_r, we have to check the satisfaction of $X \rightarrow Y$ on each relation $r_{j,j+1}$, with $j \in \mathbf{TIME}$. To this end, we need an external procedure to generate, for all j, the relation $r_{j,j+1}$ and to check whether it satisfies the functional dependency.

Again, the use of the external procedure can be avoided by checking whether the following query, whose join condition constrains the tuples to merge to be valid on two consecutive time points, returns the empty set:

$$\sigma_{Y \neq Y_1}(T\_r \bowtie_{(X=X_1 \wedge (VT+1=VT_1 \vee VT=VT_1))} T\_r_1)$$

Example. Let us consider the following temporal dependency over the relation $T\_Patient$: *"every patient assuming the same drug in two consecutive time points cannot change the type of chemotherapy in these points"*.

According to Wijsen's notation, this dependency can be expressed as $\{P\text{-}Id, Drug\}\mathbf{N}\{Chemo\}$. It is satisfied by the temporal relation $T\_Patient$ at time i, if the dependency $P\text{-}Id, Drug \rightarrow Chemo$ is satisfied by the following relation $r_{i,i+1}$:

$$r_{i,i+1} = \pi_Z(\sigma_{VT=i \vee VT=i+1}(T\_Patient))$$

If we want to check whether the relation $T\_Patient$ satisfies the given dependency, we must generate the relation $r_{i,i+1}$, for every $i = 1, \ldots, 28$, and check the satisfaction of $P\text{-}Id, Drug \rightarrow Chemo$ on it. As an alternative, the problem of checking the satisfaction of $\{P\text{-}Id, Drug\}\mathbf{N}\{Chemo\}$ over $T\_Patient$ can be brought back to the problem of checking whether the following query returns the empty set:

$$\sigma_{Chemo \neq Chemo_1}(T\_Patient \bowtie_{Cond} T\_Patient_1)$$

where

$$Cond \equiv P\text{-}Id = P\text{-}Id_1 \wedge Drug = Drug_1 \wedge VT + 1 = VT_1$$

4.4 Bettini, Jajodia, and Wang's TFDs

Bettini, Jajodia, and Wang TFD is a statement of the form $X \rightarrow_{G'} Y$, where G' is a temporal granularity [1]. Such a TFD is satisfied by a temporal instance T_r of a temporal module (R, G, ϕ) if the functional dependency $X \rightarrow Y$ is satisfied, for each $i \in \mathbb{N}$ such that $G'(i) \neq \emptyset$, by the following relation:

$$r_{[s_i,e_i]} = \pi_Z(\sigma_{(VT \geq s_i \wedge VT \leq e_i)}(T\_r))$$

where Z is the set of atemporal attributes and s_i and e_i are the granules of $G(i)$ delimiting the temporal granule $G'(i)$, that is, $G'(i) = [s_i, e_i]$ at granularity G[2]. The relation $r_{[s_i,e_i]}$ consists of the tuples of T_r, defined on the set of atemporal attributes Z, valid at some time point belonging to the i-th granule of G'. In such a case, it is necessary to check the satisfaction of $X \rightarrow Y$ in all the relations $r_{[s_i,e_i]}$ associated with the non-empty granules of G'.

As an alternative, the constraint expressed by a temporal functional dependency $X \rightarrow_{G'} Y$ can be described by means of a suitable relational algebra query. Let $Gran_{G'} = (G_S, G_E)$ be a relation whose tuples represent the starting point (G_S) and the ending point (G_E) of the granules of the (finite[3]) granularity G' at the granularity G used for the relation T_r. The TFD $X \rightarrow_{G'} Y$ is satisfied on T_r if and only if the following query returns the empty set:

$$\sigma_{Y \neq Y_1}((T\_r \bowtie_{X=X_1} T\_r_1) \bowtie_{Cond} Gran_{G'})$$

where

$$Cond \equiv G_S \leq VT \leq G_E \wedge G_S \leq VT_1 \leq G_E$$

Example. Let us consider the following temporal constraint over the relation *T_Patient*: *"a patient cannot assume different quantities of the same drug within a therapy cycle"*. In Bettini, Jajodia, and Wang's formalism, it can be expressed by the TFD $P\text{-}Id, Drug \rightarrow_{ChlVPP} Qty$. This dependency is satisfied by the relation *T_Patient* if $P\text{-}Id, Drug \rightarrow Qty$ is satisfied, for each $i \in \mathbb{N}$ such that ChlVPP$(i) \neq \emptyset$, by the relation $r_{[s_i,e_i]}$ obtained from the following relational algebra expression:

$$r_{[s_i,e_i]} = \pi_Z(\sigma_{(VT \geq s_i \wedge VT \leq e_i)}(T\_Patient))$$

As an example, with respect to the *T_Patient* instance of Table 1, the time granules s_i and e_i (at the bottom granularity of days), delimiting the temporal interval related to the first cycle of treatment, are 1 and 28. Thus, the relation $r_{[1,28]}$ contains all the tuples which are valid during the first granule of the ChlVPP granularity.

Alternatively, the problem of checking the satisfaction of $P\text{-}Id, Drug \rightarrow_{ChlVPP} Qty$ over *T_Patient* can be brought back to that of checking whether the following query returns the empty set:

$$\sigma_{Qty \neq Qty_1}(Q \bowtie_{Cond} Gran_{ChlVPP})$$

[2]For sake of simplicity, we restrict ourselves to granularities without gap inside.

[3]To deal with the general case of possibly infinite granularities, we can assume that the relation $Gran_{G'}$ is suitably instantiated by an external procedure with regard to the relation to join with.

where

$$\begin{array}{rcl} Q & \leftarrow & T\_Patient \ltimes_{P\text{-}Id=P\text{-}Id_1} T\_Patient_1 \\ Cond & \equiv & G_S \leq VT \leq G_E \wedge G_S \leq VT_1 \leq G_E \end{array}$$

The TFDs with granularities proposed by Wijsen in [16] can be dealt with in a similar way.

5 Defining new TFDs

Besides providing a uniform description of the TFDs proposed in the literature, the proposed algebraic approach allows us to easily characterize further TFDs, which, at the best of our knowledge, have not been previously considered.

As an example, let us consider the following constraint: *"the quantity of a prescribed drug for a given patient cannot be changed from one administration to the following one, if the two administrations happen within 5 days"*.

Such a condition can be expressed by the dependency $P\text{-}Id,\ Drug \rightarrow_5 Qty$, whose satisfaction can be imposed by constraining the following query to return the empty set:

$$\sigma_{(Qty \neq Qty_1 \wedge VT - VT_1 \leq 5)}(Q - \pi_W(Q \ltimes_{Cond} S))$$

where

$$\begin{array}{rcl} W & \equiv & (\mathsf{att}(T\_Patient) \cup \mathsf{att}(T\_Patient_1)) \\ Q & \leftarrow & T\_Patient \ltimes_{P\text{-}Id=P\text{-}Id_1 \wedge Drug=Drug_1} T\_Patient_1 \\ S & \leftarrow & T\_Patient_2 \ltimes_{P\text{-}Id_2=P\text{-}Id_3 \wedge Drug_2=Drug_3} \\ & & T\_Patient_3 \end{array}$$

and

$$\begin{array}{rcl} Cond & \equiv & P\text{-}Id = P\text{-}Id_2 \wedge Drug = Drug_2 \wedge \\ & & VT = VT_2 \wedge VT_1 > VT_3 \end{array}$$

The query allows one to correlate the tuples related to consecutive assumptions of the same drug for the same patient (difference and join operations). The select condition checks whether two consecutive assumptions happen within 5 days ($VT - VT_1 \leq 5$) and have different quantities for the associated drug ($Qty \neq Qty_1$). If this is the case, then the temporal dependency $P\text{-}Id,\ Drug \rightarrow_5 Qty$ is not satisfied.

This constraint represents a temporal functional dependency which cannot be expressed by the proposed TFDs. On the one hand, those devoid of granularities deal either only with tuples valid on consecutive time points or separately on tuples valid at each time point or globally on all the tuples valid at every time point. On the other hand, TFDs with multiple granularities consider sets of tuples according to a global time partition, not related to the distance between two consecutive tuples. As a matter of fact, to express the proposed dependency we would need a set of TFDs with multiple granularities. As an example, the proposed constraint can be expressed by a set of 5 TFDs in the formalism of Bettini et al. [1]. Each one of the 5 TFDs is of the form $P\text{-}Id, Drug \rightarrow_{G_i} Qty$, where i ranges from 1 to 5. The 5 granularities are defined as having granules of size 5 days and are such that each granule $G_{i+1}(j)$ starts one day after the starting of the granule $G_i(j)$.

As a further example, let us consider the following constraint: *"the quantities of a prescribed drug must be repeated for any chemotherapy every seven days"*.

In this case, TFDs with multiple granularities cannot help us, because we are constraining corresponding tuples of different groups of seven days (i.e., granules). Such a dependency can be expressed as $Chemo,\ Drug \rightarrow^{\Box 7} Qty$, and its satisfaction can be imposed by constraining the following query to return the empty set:

$$\sigma_{(Qty \neq Qty_1 \wedge VT - VT_1 = 7)}(Q - \pi_W(Q \ltimes_{Cond} S))$$

where

$$\begin{array}{rcl} W & \equiv & (\mathsf{att}(T\_Patient) \cup \mathsf{att}(T\_Patient_1)) \\ Q & \leftarrow & T\_Patient \ltimes_{Chemo=Chemo_1 \wedge Drug=Drug_1} \\ & & T\_Patient_1 \\ S & \leftarrow & T\_Patient_2 \ltimes_{Chemo_2=Chemo_3 \wedge Drug_2=Drug_3} \\ & & T\_Patient_3 \end{array}$$

and

$$\begin{array}{rcl} Cond & \equiv & Chemo = Chemo_2 \wedge Drug = Drug_2 \wedge \\ & & VT = VT_2 \wedge VT_1 > VT_3 \end{array}$$

6 Conclusions

In this paper, we have shown that the problem of checking the satisfaction of TFDs can be reduced to the problem of establishing whether the evaluation of suitable relational algebra expressions returns the empty set. This approach allows us to manage in a uniform way the different TFDs proposed in the literature and to express new meaningful TFDs.

We are currently working on the problem of establishing the complexity of checking for emptiness the resulting relational algebra expressions on a given database. It is well-known that the data complexity of relational algebra queries is in LOGSPACE with respect to the database size [12]. Efficient ad-hoc algorithms can be studied for the class of algebraic expressions needed to express relevant TFDs.

Besides the model-checking problem we addressed in this paper, a more general issue is to verify whether there exists a database satisfying a given expression (satisfiability checking). The general problem of checking temporal constraints has been extensively studied in the literature [2, 3, 4, 13] both for first-order temporal logic (FOTL) and for two-sorted classical first-order logic. In particular, in [13] Lipeck and Saake identify a restricted class of FOTL formulae, often called *biquantified formulae*, which includes formulae with only future temporal operators and admits only a restricted form of quantification defined as follows. Quantifiers can be either *external* (not in the scope of any temporal connective) or *internal* (no temporal connectives in their scope). In the considered class of formulae, all external quantifiers are universal.

In [2] Chomicki analyzes the problem of checking temporal logic constraints on the basis of the notion of *satisfaction constraints on finite histories*. The considered class of logical formulae, which allow one to capture a significant set of temporal integrity constraints, is the class of the *safety* formulae [3] (intuitively, a safety formula says that "nothing bad ever happens"). A violation of a safety formula can be detected as a violation of its satisfaction in some finite history of the database; for these reasons, this class of formulae are particularly suitable for checking constraints. The notion of *potential constraint satisfaction* [4] describes the fact that an integrity constraint is potentially satisfied at instant *t* if the finite history of the database (i.e. the sequence of instances up to the current one) can be extended to an infinite history that satisfies it. In [3] Chomicki and Niwinski prove that the problem of potential constraint satisfaction for biquantified safety formulae, with no internal quantifiers, is decidable in exponential time.

We are exploring the possibility of automatically translating our relational algebra expressions into biquantified formulae. By a preliminary analysis of TFDs, we have been able to prove that some of them can be easily expressed by a biquantified formula. As an example, Wijsen's DFDs of the form $X\mathbf{N}Y$ can be expressed by the biquantified FOTL formula $\forall X, Y, Y_1((r(X,Y) \wedge \bigcirc r(X,Y_1)) \rightarrow Y = Y_1)$, where $\bigcirc$ is the temporal operator "*next*" [5].

Finally, we would like to compare the expressiveness of the proposed (algebraic) approach with that of the logical framework for TFDs outlined in [8], based on an extension of the notion of labeled linear time structure for time granularity introduced in [6].

References

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

[2] J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems*, 20(2):149–186, 1995.

[3] J. Chomicki and D. Niwinski. On the feasibility of checking temporal integrity constraints. *Journal of Computer and System Sciences*, 51(3):523–535, 1995.

[4] J. Chomicki and G. Saake. *Logics for Databases and Information Systems*. Kluwer Academic Publishers, 1998.

[5] J. Chomicki and D. Toman. Temporal logic in information systems. In *Logics for Databases and Information Systems* [4], pages 31–70.

[6] C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004.

[7] C. Combi and A. Montanari. Data models with multiple temporal dimensions: completing the picture. In *Proc. of the 13th Conference on Advanced Information Systems Engineering*, volume 2068 of *LNCS*, pages 187–202, 2001.

[8] C. Combi and R. Rossato. Temporal functional dependencies with multiple granularities: a logic based approach. In *Proc. of the 15th Conference on Database and Expert System Applications*, volume 3180 of *LNCS*, pages 864–873, 2004.

[9] C.D. Weekes et al. Hodgkin's disease in the elderly: Improved treatment outcome with a doxorubicin-containing regimen. *Journal of Clinical Oncology*, 20(4):1087–1093, 2002.

[10] C.S. Jensen and R. T. Snodgrass. Temporally enhanced database design. In M.P. Papazoglou, S. Spaccapietra, and Z. Tari, editors, *Advances in Object-Oriented Data Modeling*, pages 163–193. MIT Press, 2000.

[11] C.S. Jensen, R.T. Snodgrass, and M.D. Soo. Extending existing dependency theory to temporal databases. *IEEE TKDE*, 8(4):563–581, 1996.

[12] P. C. Kanellakis. *Elements of Relational Database Theory*, volume B: Formal Models and Sematics of *Handbook of Theoretical Computer Science*, pages 1073–1156. Elsevier and MIT Press, 1990.

[13] U.W. Lipeck and G. Saake. Monitoring dynamic integrity constraints based on temporal logic. *Information Systems*, 12(3):255–269, 1987.

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

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

[16] J. Wijsen. Temporal FDs on complex objects. *ACM TODS*, 24(1):127–176, 1999.

[17] J. Wijsen, J. Vandenbulcke, and H. Olivie. On time-invariance and synchronism in valid-time relational databases. *Journal of Computing and Information*, pages 1192–1206, 1994.