Logical Design for Temporal Databases with Multiple Temporal Types

X. Sean Wang[*,‡], Claudio Bettini[*,‡], Alexander Brodsky[†], Sushil Jajodia[*,†]

ISSE Technical Report: ISSE-TR-94-111

ABSTRACT

The purpose of good database logical design is to eliminate data redundancy and insertion and deletion anomalies. In order to achieve this objective for temporal databases, the notions of *temporal types* and *temporal functional dependencies* (TFDs) are introduced. A temporal type is a monotonic mapping from ticks of time (represented by positive integers) to time sets (represented by subsets of reals) and is used to capture various standard and user-defined calendars. A TFD is a proper extension of the traditional functional dependency and takes the form $X \longrightarrow_\mu Y$, meaning that there is a unique value for Y during one tick of the temporal type μ for one particular X value. An axiomatization for TFDs is given. Since a finite set of TFDs usually implies an infinite number of TFDs, we introduce the notion of and give an axiomatization for a *finite closure* to effectively capture a finite set of implied TFDs that are essential to the logical design. Temporal normalization procedures with respect to TFDs are then given. More specifically, temporal Boyce-Codd normal form (TBCNF) that eliminates all data redundancies, and temporal third normal form (T3NF) that allows dependency preservation, are defined. Both normal forms are proper extensions of their traditional counterparts, BCNF and 3NF. Decomposition algorithms are presented that give lossless TBCNF decompositions and lossless, dependency preserving, T3NF decompositions.

---

[*]Partly supported by an ARPA grant, administered by the Office of Naval Research under grant number N0014-92-J-4038.

[†]Department of Information and Software Systems Engineering, George Mason University, Fairfax, VA 22030. Email: {xywang,brodsky,jajodia}@isse.gmu.edu.

[‡]Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milano, Italy. Email: bettini@dsi.unimi.it. Part of the work was performed while this author was visiting GMU.

Contents

List of Figures

1 Introduction

In the database area a large body of knowledge has been developed for addressing the problem of *logical design*:

> Given a body of data and constraints on the data to be represented in a database, how do we decide on a suitable logical structure for these data?

In the relational context, the problem of logical design can be restated as follows: How do we produce a database scheme (a collection of relation schemes) with certain desirable properties? Central to the design of database schemes is the idea of a *data dependency* which is a constraint on the allowable relations corresponding to a relation scheme. *Functional dependencies* (FDs) are widely used dependencies in the framework of logical design (cf. [9]).[1] A collection of known functional and other dependencies serve as input to the database design.

The purpose of good database design is to avoid the problems of *data redundancy*, *potential inconsistency*, *insertion anomalies*, and *deletion anomalies*. Consider a relation with attributes `AcctNo`, `Bank-branch` and `Bank-address` and FDs `AcctNo` $\rightarrow$ `Bank-branch` and `Bank-branch` $\rightarrow$ `Address`. Although each bank branch has a unique address (since `Bank-branch` $\rightarrow$ `Address` holds), the repetition of the address of a bank branch with every account in that branch is redundant in the relation. If, by mistake, we update the address of a branch in one tuple, but forget to do so in another, inconsistency arises. Furthermore, since `AcctNo` is the primary key, we cannot insert an address for a new bank branch that does not currently have at least one account. Finally, if the last account in a bank branch is deleted, we unintentionally lose track of its address. To resolve these problems, the relation (`AcctNo, Bank-branch, Address`) must be decomposed into two relations: (`AcctNo, Bank-branch`) with the FD `AcctNo` $\rightarrow$ `Bank-branch` and (`Bank-branch, Address`) with the FD `Bank-branch` $\rightarrow$ `Address`. In addition to being free of redundancy related problems cited above, the decomposition has several very desirable properties: (1) it is *lossless* in the sense that the original relation can be reconstructed by taking the natural join of the two projections and, thus, no information is lost; (2) the decomposition *preserves* the FDs in the sense that the FDs associated with the schemes in the decomposition are equivalent (identical in our example) to those associated with the original scheme; and (3) the two schemes are in *Boyce-Codd normal form* (BCNF), the strongest normal form in terms of FDs as the only dependencies. We should note that it is not always possible to derive a lossless, FD preserving

[1] Other dependencies such as *multivalued dependencies*, *join dependencies* and *tuple-generating dependencies* have also been considered [9], but they are outside the scope of our discussion here.

decomposition such that all its schemes are in BCNF; a lossless, FD preserving decomposition into schemes in the *third normal form* (3NF), which is slightly weaker than BCNF, can always be achieved (cf. [9]).

Temporal Dimension of Logical Design

The introduction of time adds a new dimension to the normalization problem. To illustrate, consider a temporal relation `ACCOUNTS` that records for each bank transaction the account number (`AcctNo`), transaction amount (`Amount`), account balance (`Balance`), accumulated interest for the month (`AccumInt`), and time of the transaction (`Time`). Accumulated interest is calculated at the annual rate of 5% every day, at the beginning of each day (but not accrued to the account until the end of the month). Therefore, the value of `AccumInt` does not change within each day, although it may change from one day to the next. The values of `Time` are timestamps consisting of the date (month/day/year) concatenated with the time of the day (up to seconds) of the transaction. We assume that for each account, every transaction is assigned a unique timestamp. A typical instance of `ACCOUNTS` is shown in Figure 1.

AcctNo	Amount	Balance	AccumInt	Time
1001	+1000	1000	0.00	3/3/93:09:01:00
1001	−500	500	0.14	3/4/93:10:01:55
1001	+200	700	0.14	3/4/93:11:00:00
1001	−315	385	0.14	3/4/93:12:19:03
1001	−255	130	0.14	3/4/93:18:00:00
1001	−10	120	0.19	3/7/93:09:00:00
1001	+100	220	0.19	3/7/93:12:01:40

Figure 1: A typical instance of the relation `ACCOUNTS`

Clearly, `ACCOUNTS` contains redundant information, since the same value for `AccumInt` is repeated several times within one day. Since the only FD associated with `ACCOUNTS` is `AcctNo, Time` $\rightarrow$ `Amount, Balance, AccumInt`, the scheme is in BCNF, in spite of the evident redundancy. The source of the redundancy here is that `AccumInt` for a particular `AcctNo` does not change within the same day. Since the underlying time in `ACCOUNTS` is second, not day, the above constraint cannot be captured by a traditional FD. Clearly, the structure of time must be taken into account if we wish to meet the objectives of logical design.

The central idea in our work is that we incorporate multiple *temporal types* in the definition of temporal functional dependencies and, consequently, in the definition of temporal normal forms, temporal lossless decomposition property, and dependency preservation property. Our design methodology based on these concepts will require that `ACCOUNTS` be decomposed into two relations `TRANSACTION-INFO` and `ACCUM-INTEREST`, as shown in Figure 2. Note that the `TRANSACTION-INFO` relation is stored in terms of `second` and `ACCUM-INTEREST` relation is stored in terms of `day` in Figure 2. Clearly both relations are free of data redundancy and other related anomalies.

AcctNo	Amount	Balance	Time
1001	+1000	1000	3/3/93:09:01:00
1001	−500	500	3/4/93:10:01:55
1001	+200	700	3/4/93:11:00:00
1001	−315	385	3/4/93:12:19:03
1001	−255	130	3/4/93:18:00:00
1001	−10	120	3/7/93:09:00:00
1001	+100	220	3/7/93:12:01:40

(a) `TRANSACTION-INFO`

AcctNo	AccumInt	Day
1001	0.00	3/3/93
1001	0.14	3/4/93
1001	0.19	3/7/93

(b) `ACCUM-INTEREST`

Figure 2: A decomposition of the instance in Figure 1

It is important to note that the result of the decomposition that eliminates data redundancy may have to include temporal types that, unlike in the above example, do not appear in the initial temporal schemes and constraints. Such would be the case if we had `week` and `month` instead of `second` and `day` in the above example.

Contributions

We introduce a general notion of a *temporal type* which is, intuitively, a mapping from *ticks of time* (represented by positive integers) to *time sets* (represented by subsets of reals). By definition, this mapping is *monotonic*, that is, time sets corresponding to larger *ticks* must have larger values. Thus various time units (days, weeks, months, years) of different calendars can be viewed as temporal types, as well as user defined types such as library opening hours, class meetings and so on.

We next present our notion of *temporal functional dependencies* (TFDs) which requires that underlying temporal type be designated. An example of a TFD is AcctNo $\longrightarrow_{\texttt{day}}$ AccumInt which states that AccumInt of a specific AcctNo cannot change within a day. Note that a TFD may designate any temporal type (day in the TFD), independent of the temporal type used to store the corresponding temporal relation (it could be stored, e.g., in seconds or hours).

A *sound* and *complete* axiomatization for TFDs is developed. Unfortunately, unlike in the case of traditional FDs, there is usually an infinite number of TFDs implied from a given finite set F. To overcome this problem, we introduce the notion of a *finite closure* of TFDs, and develop sound and complete axiomatization to effectively compute finite closures.

The property of *lossless decomposition* is similar to the traditional one in that it is possible to reconstruct the original temporal relation from its projections. However, the traditional natural join operation turns out to be insufficient for this purpose (e.g., it is not clear how we should join the relations in Figure 2 to recover the original ACCOUNTS relation in Figure 1), and we need to introduce new *temporal join, projection, and union* operators that incorporate temporal types, which are then used in the definition of *temporal lossless decomposition.*

The central part of the paper gives the definitions of *temporal BCNF* (TBCNF) and *temporal 3NF* (T3NF) and algorithms for achieving TBCNF and T3NF decompositions. Intuitively, a scheme R with temporal type μ is in TBCNF if every non-trivial TFD $X \longrightarrow_{\nu} Y$ is (1) a *temporal superkey* of the scheme (this is analogous to the requirement for traditional BCNF), and (2) there is no temporal redundancy due to the fact that two different time ticks of μ belong to the same time tick of ν.

To understand the motivation behind the second condition, consider two temporal tuples t and t' that agree on X and whose time ticks in terms of μ belong to the same time tick of ν. Because of $X \longrightarrow_{\nu} Y$ we can conclude that the Y values of t and t' must also coincide. Since this information can be implied, we have redundancy that we would like to avoid.

A decomposition algorithm is given that renders lossless decomposition of any temporal scheme into schemes that are in TBCNF. We discuss the preservation of temporal functional dependencies in the decompositions of the temporal schemes. Analogous to the traditional relational theory, the decomposition of temporal schemes into TBCNF may not preserve TFDs. Therefore, we give the definition of temporal 3NF (T3NF), and present an algorithm that provides lossless, dependency-preserving, T3NF decomposition.

Related Work

There has been work on normal forms for temporal relations, including *first temporal normal form* [8], *time normal form* [6], P and Q *normal forms* [5], and temporal extensions of traditional normal forms in [3]. However, none of these takes the structure of time into account and, therefore, cannot treat the redundancy related problems such as those in the `ACCOUNTS` example.

The work in [3] is closest to ours in that we both consider dependencies and normalization for a general, not some specialized, temporal data model. Moreover, the temporal extensions in both satisfy the following properties which are fundamental to the traditional normalization theory [3]: (1) dependencies are intentional, not extensional properties, (2) normal forms are properties defined solely in terms of the dependencies that are intended to be satisfied, (3) normal forms are properties of stored (base) relations only, and (4) FDs and normal forms are defined independently of the representation of a relation. The normal forms in [8, 6, 5] do not satisfy one or more of these properties [3].

Conceptually, [3] views each temporal relation as a collection of snapshot relations. Since each snapshot is a relation in the conventional (nontemporal) relational model, the usual notions of FDs, multivalued dependencies and normal forms are applied. Thus, for example, according to [3] a temporal relation satisfies an FD if each snapshot satisfies the given FD. Likewise, a temporal relation is in Boyce-Codd temporal normal form (BCTNF) if each snapshot is in BCNF. Returning to the `ACCOUNTS` relation in Figure 1, since every transaction for an account is assigned a unique timestamp, FD `AcctNo` $\rightarrow$ `Amount Balance AccumInt` is valid in every snapshot and, therefore, `ACCOUNTS` is in BCTNF according to [3], in spite of the evident redundancy and other related anomalies in `ACCOUNTS`.

Organization of the Paper

The rest of the paper is organized as follows. In Section 2, the temporal types are defined and temporal modules are reviewed. TFDs are introduced in Section 3. Also presented in Section 3 are the sound and complete axioms and finite closures of TFDs. Temporal normalization is discussed in Section 4, in which lossless decompositions and related notions are introduced. TBCNF and its decomposition algorithm is presented in Section 5. Dependency preserving decompositions are studied in Section 6. And finally, T3NF and its decomposition algorithm are presented in Section 7. Section 8 concludes the paper with some discussion. The Appendix provides proofs for several theorems.

2 Temporal Types and Modules

2.1 Temporal types

Before we can incorporate multiple temporal types into the logical design of temporal databases, we first need to formalize the notion of a temporal type.

We assume that there is an underlying notion of absolute time, represented by the set of the real numbers.[2] In the following we will denote by $\mathcal{R}$ the set of all real numbers.

Definition (Temporal type)

A *temporal type* is a mapping μ from the set of the positive integers (the *time ticks*) to $2^{\mathcal{R}}$ (the set of absolute time sets) such that for all positive integers i and j with $i < j$, the following two conditions are satisfied:

1. $\mu(i) \neq \emptyset$ and $\mu(j) \neq \emptyset$ imply that each real number in $\mu(i)$ is less than all real numbers in $\mu(j)$, and
2. $\mu(i) = \emptyset$ implies $\mu(j) = \emptyset$.

Property (1) states that the mapping must be *monotonic*. Property (2) disallows an empty set to be the value of a mapping for a certain time tick unless the empty set will be the value of the mapping for all subsequent time ticks.

Intuitive temporal types, e.g., `day`, `month`, `week` and `year`, satisfy the above definition. For example, we can define a special temporal type `year` starting from year 1800 as follows: `year`(1) is the set of absolute time set (an interval of reals) corresponding to the year 1800, `year`(2) is the set of absolute time set corresponding to the year 1801, and so on. Note that the sets in the type `year` are consecutive intervals; however, this does not have to be the case for all types. Leap years, which are not consecutive intervals, also constitute a temporal type. If we take 1892 as the first leap year, then `leap-year`(2) corresponds to 1896, `leap-year`(3) corresponds to 1904,[3] `leap-year`(4) corresponds to 1908, and so on. We can also represent a finite collection of "ticks" as a temporal type as well. For example, to specify the year 1993, we can use the temporal type T such that $T(1)$ is the set of absolute time corresponding to the year 1993, and $T(i) = \emptyset$ for each $i > 1$.

The definition also allows temporal types in which ticks are mapped to more than a single interval. This is a generalization of most previous definitions of temporal types. As an example, consider the

[2] In fact, any infinite set with a total ordering can serve as the absolute time; reals, rationals and integers are examples.

[3] Note 1900 is not a leap year

type `business-month`, where every business month is a union of all business days in a month (i.e., excluding all Saturdays and Sundays). In this case, more than one single interval is in one tick.

In a realistic system, we should distinguish names of temporal types, like `day`, used by users or system designers from the mapping they denote. It is possible that in a system different names can be used for the same mapping. For example, `day` and `giorno` (which is the Italian word for `day`) denote the same mapping. However, for simplicity, in this paper we assume that different symbols used for temporal types denote different mappings.

It is important to emphasize that a real system can only treat infinite types that have finite representations. Various periodical descriptions, e.g., [4, 7], are possible but outside the scope of this paper.

There is a natural relation among temporal types as given below:

Definition Let μ_1 and μ_2 be temporal types. Then μ_1 is said to be *finer than* μ_2, denoted $\mu_1 \preceq \mu_2$, if for each i, there exists j such that $\mu_1(i) \subseteq \mu_2(j)$.

This "finer than" relation is essential for temporal FDs. In the `ACCOUNTS` relation, since the accumulated interest does not change in a day, it does not change in any hour during the day. (Note that `hour` is finer than `day`.) In fact, the interest will not change in terms of any temporal type that is finer than `day`. We will discuss this issue further in Section 3. We only want to note here that there are infinite number of temporal types that are finer than `day`.

In the rest of the paper conditions like $\mu_1(i) \subseteq \mu_2(j)$ are often expressed as "tick j of μ_2 *covers* tick i of μ_1." The notation $\mu \prec \nu$ will be used for a *strictly finer than* relation: i.e., $\mu \prec \nu$ if $\mu \preceq \nu$ and $\mu \neq \nu$.

Consider now the properties of the *finer than* relation. By definition, $\mu \preceq \mu$ for each time unit μ. Also, if $\mu_1 \preceq \mu_2$ and $\mu_2 \preceq \mu_1$ then $\mu_1 = \mu_2$. Furthermore, the finer-than relation is obviously transitive. Thus, $\preceq$ is a partial order. The relation $\preceq$ is not a total order since, for example, `week` and `month` are incomparable (i.e., `week` is not finer than `month`, and `month` is not finer than `week`). There exists a unique least upper bound of the set of all temporal types denoted by μ_{Top}, and a unique greatest lower bound, denoted by μ_{Bottom}. These top and bottom elements are defined as follows: $\mu_{\text{Top}}(1) = \mathcal{R}$ and $\mu_{\text{Top}}(i) = \emptyset$ for each $i > 1$, and $\mu_{\text{Bottom}}(i) = \emptyset$ for each positive integer i. Moreover, it is easily seen that for each pair of temporal types μ_1, μ_2, there exist a unique least upper bound $lub(\mu_1, \mu_2)$ and a unique greatest lower bound $glb(\mu_1, \mu_2)$ of the two types, with respect to $\preceq$. That is, the set of all temporal types forms a lattice with respect to $\preceq$.

Proposition 1 The set of all temporal types is a lattice with respect to the finer than relation.

2.2 Temporal modules

Our discussion of logical design for temporal databases is in terms of temporal modules that were introduced in [12] to provide a unified interface for accessing different underlying temporal information systems. Thus, the temporal module concept is rather general; the concepts and the results of this paper are readily translated to in terms of other temporal data models. Temporal modules defined in this paper are simplified but equivalent versions of extended temporal modules of [12], explained below.

We assume there is an infinite set of attributes. For each attribute A, there exists an infinite set of values called domain of A, denoted $dom(A)$. Each finite set R of attributes is called a *relation scheme*. A relation scheme $R = \{A_1, \ldots, A_n\}$ is usually written as $\langle A_1, \ldots, A_n \rangle$. For relation scheme R, let $Tup(R)$ denote the set of all mappings t, called *tuples*, from R to $\bigcup_{A \in R} dom(A)$ such that for each A in R, $t(A)$ is in $dom(A)$. A tuple of relation scheme $\langle A_1, \ldots, A_n \rangle$ is usually written as $\langle a_1, \ldots, a_n \rangle$, where a_i is in $dom(A_i)$ for each $1 \leq i \leq n$. We are now ready to define temporal module schemes and temporal modules.

Definition (Temporal module scheme and temporal module)
A *temporal module scheme* is a pair (R, μ), where R is a relation scheme and μ a temporal type. A *temporal module* is a triple (R, μ, ϕ), where (R, μ) is a temporal module scheme and ϕ is a function, called *time windowing* function from $\mathbf{N}$ to $2^{\mathbf{Tup}(R)}$ such that $\phi(i) = \emptyset$ for each i with $\mu(i) = \emptyset$.

Intuitively, the time windowing function ϕ in a temporal module (R, μ, ϕ) gives the tuples (facts) that hold at time tick i in temporal type μ. This is a generalization of many temporal models appeared in the literature.

The temporal module (R, μ, ϕ) is said to be *on* (R, μ) and to be *in terms of* μ. Temporal modules are also denoted by symbol $\mathtt{M}$, possibly subscripted. For each temporal module $\mathtt{M}$, we denote its relation scheme, temporal type, and windowing function by $R_{\mathtt{M}}$, $\mu_{\mathtt{M}}$ and $\phi_{\mathtt{M}}$, respectively. For convenience, in temporal module examples, instead of the positive integers we will sometimes use an equivalent domain. For instance, the set of expressions of the form $3/3/93 : 09 : 01 : 00$ will serve as such a domain.

Example 1 We view the temporal relation `ACCOUNTS` given in the introduction as a temporal module with (`ACCOUNTS`, `second`), where `ACCOUNTS` = ⟨`AcctNo`, `Amount`, `Balance`, `AccumInt`⟩, as its scheme. The relation in Figure 1 corresponds to the time windowing function ϕ defined as follows:

$$\phi(3/3/93:09:01:00) = \{\langle\ 1001,\ +1000,\ 1000,\ 0.00\ \rangle\}$$
$$\phi(3/4/93:10:01:55) = \{\langle\ 1001,\ -500,\ 500,\ 0.14\ \rangle\}$$
$$\phi(3/4/93:11:00:00) = \{\langle\ 1001,\ +200,\ 700,\ 0.14\ \rangle\}$$
$$\phi(3/4/93:12:19:03) = \{\langle\ 1001,\ -315,\ 385,\ 0.14\ \rangle\}$$
$$\phi(3/4/93:18:00:00) = \{\langle\ 1001,\ -255,\ 130,\ 0.14\ \rangle\}$$
$$\phi(3/7/93:09:00:00) = \{\langle\ 1001,\ -10,\ 120,\ 0.19\ \rangle\}$$
$$\phi(3/7/93:12:01:40) = \{\langle\ 1001,\ +100,\ 220,\ 0.19\ \rangle\}$$

□

Two equivalent query languages, TM-calculus [12] and TM-algebra [11], have been proposed to frame queries on temporal modules.

3 Temporal Functional Dependencies

Relations in relational databases are traditionally used to store "static" information, or only the "current" information. An FD $X \to Y$ states that whenever a relation has two tuples that agree on attributes X, then they agree on attributes Y also. (See [9] for formal definitions.) As an example, consider a relation scheme, called `FACULTY`, that records for each faculty member, the social security number (`SSn`), name (`Name`), rank (`Rank`), and department (`Dept`). FD `SSn` $\to$ `Rank` states that each faculty member's rank is unique, even though he or she may serve in more than one department at the same time.

In a temporal database, information becomes dynamic. An FD that is valid in the "current" relation may no longer be valid in the corresponding temporal relation if the traditional definition of FDs is used without change. This will be the case if `FACULTY` were a temporal relation since it is likely that `FACULTY` will contain two tuples, one stating that a particular faculty is an Assistant Professor at one time, but an Associate Professor at a different time. We will extend the traditional notion of FDs that will not only permit these possibilities, but enable us to model additional constraints such as "a faculty member's rank does not change during an academic year" also.

Definition (Temporal functional dependency)
Let X and Y be (finite) sets of attributes and μ a temporal type such that $\mu(i) \neq \emptyset$ for some i. Then $X \longrightarrow_\mu Y$ is called a *temporal functional dependency (TFD).*

Intuitively, a TFD $X \longrightarrow_\mu Y$ states that whenever two tuples, holding on time ticks covered by the same time tick in μ, agree on X, they must also agree on Y. Thus, the TFD `SSn` $\longrightarrow_{\texttt{ay}}$ `Rank`, where `ay` is the temporal type of academic years, expresses the fact that a faculty's rank cannot change during an academic year. We now formally define the above notion of satisfaction. In order to simplify the notation throughout this paper, we will use $\mu(i_1, \ldots, i_k)$ to denote $\bigcup_{1 \leq j \leq k} \mu(i_j)$.

Definition (Satisfaction of TFD)
A TFD $X \longrightarrow_\nu Y$ is satisfied by a temporal module $\mathtt{M} = (R, \mu, \phi)$ if for all tuples t_1 and t_2 and positive integers i_1 and i_2, the following three conditions imply $t_1[Y] = t_2[Y]$:

(a) $t_1[X] = t_2[X]$,

(b) t_1 is in $\phi(i_1)$ and t_2 is in $\phi(i_2)$, and

(c) there exists j such that $\mu(i_1, i_2) \subseteq \nu(j)$.

For example, the temporal module corresponding to Figure 1 satisfies the TFD `AcctNo` $\longrightarrow_{\mathtt{day}}$ `AccumInt`. The temporal module also satisfies `AcctNo` $\longrightarrow_{\mathtt{second}}$ `Balance`. However, the temporal module does *not* satisfy `AcctNo` $\longrightarrow_{\mathtt{day}}$ `Balance`.

The definition implies that the TFD $X \longrightarrow_\nu Y$ is always satisfied by the temporal module (R, μ, ϕ) if μ does not have two different ticks that are covered by a single tick of ν (since condition (c) in the definition will not be satisfied).

Let (R, μ) be a temporal module scheme with a set F of TFDs defined on (R, μ). Only temporal modules that satisfy F are considered *feasible* or *allowed*. Thus the set F determines the set of feasible modules.

Example 2 The temporal module reported in Example 1 satisfies the TFD `AcctNo` $\longrightarrow_{\mathtt{second}}$ `Amount`, `Balance`, `AccumInt`. In this case both the temporal module and the TFD are defined in terms of the same temporal type (`second`). However, the same module satisfies the TFD `AcctNo` $\longrightarrow_{\mathtt{day}}$ `AccumInt` that is defined in terms of a different temporal type. The notion of temporal FDs introduced in [3] allows only TFDs in terms of the same temporal type of the module. □

Our notion of TFDs can also be used to express FDs that always hold, independent of time. For example, we can express the FD "each person has only one biological father (regardless of time)" using the TFD `Name` $\longrightarrow_{\mu_{\mathrm{Top}}}$ `B_Father`. This TFD says that whenever two facts t_1 and t_2 with $t_1[X] = t_2[X]$ are valid in any temporal module, then $t_1[Y] = t_2[Y]$, independent of the temporal type.

3.1 Inference axioms for TFDs

As in the case of traditional FDs, inference axioms to derive all TFDs that logically follow from a set of TFDs are important. The inference axioms given below include not only the temporal analogs of Armstrong's axioms [9], but axioms that reflect the relationships among temporal types also. In Example 1, since the values for accumulated interest values do not change in a day, they do not change in any hour of the day. In general, they do not change in any tick of any temporal type that is finer than `day`. This is captured by the inference rule: if $X \longrightarrow_{\mu} Y$ and $\nu \preceq \mu$, then $X \longrightarrow_{\nu} Y$.

An intuitive conjecture would be that the above rule along with the temporal analogs of Armstrong's axioms will constitute a complete axiomatization. This turns out to be false. Consider, for example, two TFDs $X \longrightarrow_{\mathtt{y1}} Y$ and $X \longrightarrow_{\mathtt{y2}} Y$, where `y1` is the temporal type corresponding to years before 1990, and `y2` is the temporal type corresponding to years 1990 and beyond. Taken together, these two TFDs say that $X \longrightarrow_{\mathtt{year}} Y$. However, `year` is not finer than either `y1` or `y2`.

In order to capture such inferences, we define the notion that a temporal type is *collectively finer* than a set of temporal types. The temporal type `year` will be collectively finer than the set of types $\{\mathtt{y1}, \mathtt{y2}\}$ because each tick of the type `year` is covered by (i.e. contained in) a tick of either `y1` or `y2`. Formally,

Definition We say that a temporal type ν is *collectively finer than* a set $\{\mu_1, \ldots, \mu_n\}$ of temporal types, denoted $\nu \preceq_C \{\mu_1, \ldots, \mu_n\}$, if for each positive integer i, there exist $1 \leq k \leq n$ and a positive integer j such that $\nu(i) \subseteq \mu_k(j)$

Note that $\mu \preceq \mu_1$ implies $\mu \preceq_C \{\mu_1, \mu_2\}$ for any μ_2.

Inference axioms for TFDs are given next.

Four Inference Axioms for TFDs:

1. *Reflexivity:* If $Y \subseteq X$, then $X \longrightarrow_{\mu} Y$ for each temporal type μ.
2. *Augmentation:* If $X \longrightarrow_{\mu} Y$, then $XZ \longrightarrow_{\mu} XZ$.
3. *Transitivity:* If $X \longrightarrow_{\mu} Y$ and $Y \longrightarrow_{\mu} Z$, then $X \longrightarrow_{\mu} Z$.
4. *Descendability:* If $X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_n} Y$ with $n \geq 1$ then $X \longrightarrow_{\mu} Y$ for each μ with $\mu \preceq_C \{\mu_1, \ldots, \mu_n\}$.

The first three axioms are temporal analogs of the Armstrong's axioms. The descendability axiom states that if one or more TFDs (with the same left and right hand sides X and Y) in terms of different types are satisfied by a temporal module $\mathtt{M}$, then a TFD (with the same X and Y) in terms of any temporal type that is finer than the set of these temporal types is also satisfied by $\mathtt{M}$. In particular, if we know that $X \longrightarrow_{\nu} Y$ is satisfied by $\mathtt{M}$, descendability ensures that for each μ with μ finer than ν, $X \longrightarrow_{\mu} Y$ is satisfied by $\mathtt{M}$. This makes intuitive sense: if the rank of a faculty cannot change within an academic year, it cannot change within a month or a day of an academic year.

Let F be a finite set of TFDs. The notion of derivation of TFDs is analogous to the one for traditional FDs. Formally,

Definition The TFD $X \longrightarrow_{\mu} Y$ is *derived from* F, denoted $F \vdash X \longrightarrow_{\mu} Y$, if there exists a *proof sequence* $f_1, \ldots, f_k$ such that (i) f_k is $X \longrightarrow_{\mu} Y$ and (ii) each f_i is a TFD either in F or obtained by using one of the four axioms on TFDs $f_1, \ldots, f_{i-1}$.

The notion of implication of TFD is also standard. Formally,

Definition The TFD $X \longrightarrow_{\mu} Y$ is *logically implied by* F, denoted $F \models X \longrightarrow_{\mu} Y$, if every temporal module $\mathtt{M}$ that satisfies each TFD in F, also satisfies $X \longrightarrow_{\mu} Y$.

Below, we establish the fact that the four axioms are *sound* (i.e., they can be used to derive only logically implied TFDs) and *complete* (i.e., they can be used to derive all logically implied TFDs). First, we have the soundness result:

Lemma 1 The four inference axioms are sound.

Proof. We must prove that for each set F of TFDs, given a TFD $X \longrightarrow_{\mu} Y$, $F \vdash X \longrightarrow_{\mu} Y$ implies $F \models X \longrightarrow_{\mu} Y$. It is sufficient to show that each derivation rule (axiom) starting from logically valid TFDs derives only logically valid TFDs. The soundness of the first three axioms is trivial. Consider the descendability axiom, i.e., $X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_n} Y$ implies $X \longrightarrow_{\mu} Y$ for each $\mu \preceq_C \{\mu_1, \ldots, \mu_n\}$. By definition, $\mu \preceq_C \{\mu_1, \ldots, \mu_n\}$ means that each tick of μ is *covered* by a tick of at least one of the $\mu_1, \ldots, \mu_n$ temporal types. Let $\mathtt{M} = (R, \nu, \phi)$ be an arbitrary module that satisfies F. Since $X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_n} Y$ are (assumed to be) logically valid they are also satisfied by $\mathtt{M}$. To prove that $\mathtt{M}$ satisfies $X \longrightarrow_{\mu} Y$ by contradiction, assume $X \longrightarrow_{\mu} Y$ is violated by $\mathtt{M}$. Thus, there exists a tick of μ, say $\mu(i)$, that covers one or more ticks in ν and the windowing function of $\mathtt{M}$ in these ticks has two tuples with the same value for X but different values for Y. Since $\mu \preceq_C \{\mu_1, \ldots, \mu_n\}$, these ticks are covered also by a tick of a μ_k among the $\mu_1, \ldots, \mu_n$. Hence, $\mathtt{M}$ does not satisfy $X \longrightarrow_{\mu_k} Y$, and this is a contradiction. Since this holds for every module satisfying F, we conclude that $X \longrightarrow_{\mu} Y$ is logically valid. □

By using the above four inference axioms, we may derive other inference rules. For example, given $X \longrightarrow_{\mu_1} Y$ and $Y \longrightarrow_{\mu_2} Z$, we may derive $X \longrightarrow_{glb(\mu_1,\mu_2)} Z$. (We call this rule the *extended transitivity* axiom.) Indeed, since $glb(\mu_1,\mu_2) \preceq \mu_1$ and $glb(\mu_1,\mu_2) \preceq \mu_2$, $X \longrightarrow_{glb(\mu_1,\mu_2)} Y$ and $Y \longrightarrow_{glb(\mu_1,\mu_2)} Z$ by descendability. By transitivity, $X \longrightarrow_{glb(\mu_1,\mu_2)} YZ$. Another rule we may derive is as follows: given $X \longrightarrow_{\mu_1} Y$ and $X \longrightarrow_{\mu_2} Z$, we have $X \longrightarrow_{glb(\mu_1,\mu_2)} YZ$. (We call this *union* axiom). To see this, we use augmentation to get $X \longrightarrow_{\mu_1} XY$ and $XY \longrightarrow_{\mu_2} YZ$ from $X \longrightarrow_{\mu_1} Y$ and $X \longrightarrow_{\mu_2} Z$, respectively. Then by the extended transitivity above, we have $X \longrightarrow_{glb(\mu_1,\mu_2)} YZ$. In summary, we have the following two additional inference axioms:

Additional Inference Axioms for TFDs:

1. *Extended Transitivity:* If $X \longrightarrow_{\mu_1} Y$ and $Y \longrightarrow_{\mu_2} Z$, then $X \longrightarrow_{glb(\mu_1,\mu_2)} Z$.

2. *Union:* If $X \longrightarrow_{\mu_1} Y$ and $X \longrightarrow_{\mu_2} Z$, then $X \longrightarrow_{glb(\mu_1,\mu_2)} YZ$.

Unlike in the case of traditional FDs, the application of the TFD inference axioms on a finite set F of TFDs may lead to an infinite number of temporal functional dependencies. This is due to reflexivity and descendability axioms. It is obvious that the reflexivity axiom gives infinite number of TFDs. However, these TFDs are trivial ones. The more serious problem is the descendability axiom. The reason why the descendability axiom gives an infinite number of TFDs is that given a type (or a set of types) there may be an infinite number of types that are (collectively) finer than the given type (or set of types). Consider, for example, $\texttt{AcctNo} \longrightarrow_{\texttt{day}} \texttt{AccumInt}$. Let $\texttt{day}_i$ be the temporal type that covers only the i-th day, i.e., $\texttt{day}_\texttt{i}(1)$ maps to the absolute time of the day i, and $\texttt{day}_\texttt{i}(j) = \emptyset$ for all $j > 1$. Then clearly, $\texttt{day}_\texttt{i} \preceq \texttt{day}$ and hence $\texttt{day}_\texttt{i} \preceq_C \{\texttt{day}\}$ for all $i \geq 1$. Therefore, by the descendability axiom, we have $\texttt{AcctNo} \longrightarrow_{\texttt{day}_i} \texttt{AccumInt}$ for all $i \geq 1$. These are infinite number of TFDs.

To overcome the problem of infinite number of logically implied TFDs, we ask the following questions: Does there exist a finite set F' of TFDs which has the property that every TFD logically implied by F can be derived from F' by just one application of the descendability axiom? Can the set F' be effectively computed? We answer both questions positively by developing *three finite inference axioms* described below.

Three Finite Inference Axioms for TFDs:

1. *Restricted Reflexivity:* If $Y \subseteq X$, then $X \longrightarrow_{\mu_{\mathrm{Top}}} Y$.

2. *Augmentation:* If $X \longrightarrow_\mu Y$, then $XZ \longrightarrow_\mu YZ$.

3. *Extended Transitivity:* If $X \longrightarrow_{\mu_1} Y$ and $Y \longrightarrow_{\mu_2} Z$, then $X \longrightarrow_{glb(\mu_1,\mu_2)} Z$.

If a TFD $X \longrightarrow_{\mu} Y$ is derived by using the three finite inference axioms from a set F of TFDs, we then say $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu} Y$. It is easily seen that if $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu} Y$, then μ is the *glb* of some temporal types appearing in F. Since F is finite, we know that there are only finite number of TFDs that are derived from F by using the three finite inference axioms. We call the set of TFDs that are derived from F by these Finite Inference Axioms as the "finite closure" of F. Formally,

Definition (Finite closure)
Let F be a set of TFDs. The *finite closure* of F, denoted $\overline{F}^{+}$, is the set of all the TFDs derivable from F by the Three Finite Inference Axioms. More formally, $\overline{F}^{+} = \{X \longrightarrow_{\mu} Y \mid F \vdash_{\mathrm{f}} X \longrightarrow_{\mu} Y\}$.

Consider, for example, $F = \{A \longrightarrow_{\mu} B, A \longrightarrow_{\nu} B\}$. By the augmentation axiom, we obtain $A \longrightarrow_{\mu} AB$ and $AB \longrightarrow_{\nu} B$ from $A \longrightarrow_{\mu} B$ and $A \longrightarrow_{\nu} B$ respectively. Then by extended transitivity, we have $A \longrightarrow_{glb(\mu,\nu)} B$. Thus, $A \longrightarrow_{glb(\mu,\nu)} B$ is in $\overline{F}^{+}$.

The Three Finite Axioms are sound since each of the three axioms is derived from the Four Inference Axioms which are sound by Lemma 1. We shall show that the Three Finite Axioms are *complete up to descendability*, i.e., if $F \models X \longrightarrow_{\mu} Y$ then there exist $X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_m} Y$ in $\overline{F}^{+}$ such that $\mu \preceq_C \{\mu_1, \ldots, \mu_m\}$.

Theorem 1 The Three Finite Axioms are sound, and complete up to descendability.

Hence, although there may be an infinite number of TFDs that are logically implied by a finite set F of TFDs, the only source of infiniteness is that there may be infinite number of temporal types that are collectively finer than several temporal types which appear in the finite closure of F.

The soundness in Theorem 1 follows directly from the soundness of the Four Inference Axioms. We postpone the completeness (up to descendability) proof of the Three Finite Axioms until after we show how it implies the completeness of the Four Inference Axioms.

Theorem 2 The Four Inference Axioms for TFDs are both sound and complete.

Proof. Soundness is provided by Lemma 1. For completeness, let F be a set of TFDs and $X \longrightarrow_{\mu} Y$ a TFD logically implied by F. By Theorem 1 we know that there exist TFDs $X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_k} Y$ in $\overline{F}^{+}$ such that $\mu \preceq_C \{\mu_1, \ldots, \mu_k\}$. By the definition of $\overline{F}^{+}$ we know that, for each $1 \leq i \leq k$, $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu_i} Y$ and hence, $F \vdash X \longrightarrow_{\mu_i} Y$. Applying the descendability axiom we obtain $F \vdash X \longrightarrow_{\mu} Y$, which concludes the completeness proof. $\Box$

We now turn to prove the completeness (up to descendability) of the Three Finite Axioms. First, we give the following auxiliary operation: For each set F of TFDs and a set $\mathcal{S}$ of real numbers, let $\pi_{\mathcal{S}}(F)$ be the set of regular functional dependencies that have "effects" on $\mathcal{S}$. Formally, let

$$\pi_{\mathcal{S}}(F) = \{X \to Y \mid \exists X \longrightarrow_{\nu} Y \in F \text{ and } \exists j\ (\mathcal{S} \subseteq \nu(j))\}$$

Clearly, $\pi_{\emptyset}(F)$ gives the "non-temporal version" of all the TFDs in F.

To prove Theorem 1, we need the following two lemmas. The first lemma formalizes an important relationship between temporal and corresponding nontemporal functional dependencies.

Lemma 2 Let F be a set of TFDs and $X \longrightarrow_{\mu} Y$ a TFD such that $F \models X \longrightarrow_{\mu} Y$. Then for all i such that $\mu(i) \neq \emptyset$, we have $\pi_{\mu(i)}(F) \models X \to Y$.

Proof. Let i be a positive integer, with $\mu(i) \neq \emptyset$, and r a (non-temporal) relation that satisfies $\pi_{\mu(i)}(F)$. We need only to show that r satisfies $X \to Y$. Let $\mathtt{M}$ be the temporal module (R, μ, ϕ), where R is the relation scheme of r and ϕ is given as follows: $\phi(i) = r$ and $\phi(j) = \emptyset$ for each $j \neq i$. [Since $\mu(i) \neq \emptyset$, $\mathtt{M}$ is well defined.] We claim that $\mathtt{M}$ satisfies F. Indeed, suppose $V \longrightarrow_{\nu} W$ is in F. If there does not exist j such that $\mu(i) \subseteq \nu(j)$, then $V \longrightarrow_{\nu} W$ is satisfied by $\mathtt{M}$. Otherwise, since there does exist j such that $\mu(i) \subseteq \nu(j)$, $V \to W$ is in $\pi_{\mu(i)}(F)$. From the fact that r satisfies $\pi_{\mu(i)}(F)$, it follows that $\mathtt{M}$ satisfies $V \longrightarrow_{\nu} W$. Thus, $\mathtt{M}$ satisfies F. By the hypothesis, $\mathtt{M}$ satisfies $X \longrightarrow_{\mu} Y$. In order to show that r satisfies $X \to Y$, let t_1 and t_2 be two arbitrary tuples in r such that $t_1[X] = t_2[X]$. We only need to show that $t_1[Y] = t_2[Y]$. By the construction of $\mathtt{M}$, t_1 and t_2 are both in $\phi(i)$. Since $\mathtt{M}$ satisfies $X \longrightarrow_{\mu} Y$, by definition, $t_1[Y] = t_2[Y]$ as desired. □

The next lemma establishes the relationship between the temporal types in a set of TFDs and those of TFDs derived from the set by using the Three Finite Inference Axioms. We will use the following notation: For each set G of TFDs, define $glb(G)$ to be the temporal type $glb(\{\nu | V \longrightarrow_{\nu} W \in G\})$. Note that $glb(G) = \mu_{\mathrm{Top}}$ if $G = \emptyset$ (see [2]).

Lemma 3 Let F be a set of TFDs and $X \longrightarrow_{\mu} Y$ a TFD. If $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu} Y$ and there is no proper subset F' of F such that $F' \vdash_{\mathrm{f}} X \longrightarrow_{\mu'} Y$ for any μ', then $\mu = glb(F)$.

Proof. Since $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu} Y$, it is easily seen that $glb(F) \preceq \mu$ by observing the Three Finite Inference Axioms. Now we show $\mu \preceq glb(F)$. Suppose by contradiction that $\mu \not\preceq glb(F)$. Hence, there exists a non-empty tick i of μ such that $\mu(i) \not\subseteq glb(F)(j)$ for all j. By the definition of $glb(F)$, it is easily seen that there exists $V \longrightarrow_{\nu} W$ in F such that $\mu(i) \not\subseteq \nu(j)$ for all j. Thus, $\pi_{\mu(i)}(F)$ is a proper subset of $\pi_\emptyset(F)$. By the soundness of the Four Inference Axioms, hence the soundness of the Three Finite Inference Axioms, we know $F \models X \longrightarrow_\mu Y$ since $F \vdash_f X \longrightarrow_\mu Y$. By Lemma 2, $\pi_{\mu(i)}(F) \vdash X \rightarrow Y$. Hence, there is a proof sequence for $X \rightarrow Y$ from $\pi_{\mu(i)}(F)$ by using the Armstrong's axioms.[4] For each Armstrong's axiom, we find a counterpart in the finite inference axioms. Also, if $Z_1 \rightarrow Z_2$ in $\pi_{\mu(i)}(F)$ is used in the proof sequence, we use $Z_1 \longrightarrow_{\nu'} Z_2$ in F where $\mu(i) \subseteq \nu'(j)$ for some j ($Z_1 \longrightarrow_{\nu'} Z_2$ in F is guaranteed by the definition of $\pi_{\mu(i)}(F)$). It is easily seen that $V \longrightarrow_\nu W$ is not used in this process. Thus, we have $(F - \{V \longrightarrow_\nu W\}) \vdash_f X \longrightarrow_{\mu'} Y$ for some μ'. This is a contradiction. Therefore, $\mu \preceq glb(F)$, and hence $\mu = glb(F)$. □

We now establish the completeness up to descendability of the Three Finite Axioms.

Proof. Suppose $F \models X \longrightarrow_\mu Y$. Since by definition, there exists i such that $\mu(i) \neq \emptyset$. By Lemma 2, we have $\pi_{\mu(i)}(F) \models X \rightarrow Y$. Since $\pi_{\mu(i)}(F) \subseteq \pi_\emptyset(F)$, we have $\pi_\emptyset(F) \models X \rightarrow Y$. We call a set G of TFDs a *support* for $X \rightarrow Y$ if $\pi_\emptyset(G) \models X \rightarrow Y$. We say that it is *minimal* if no proper subset of G is a support for $X \longrightarrow Y$. Since $\pi_\emptyset(F) \models X \rightarrow Y$, there exists at least one minimal support for $X \rightarrow Y$. We claim:

- For each minimal support F_1 of $X \rightarrow Y$, $F \vdash_f X \longrightarrow_{glb(F_1)} Y$.

Indeed, let F_1 be a minimal support for $X \rightarrow Y$. Thus, there exists a proof sequence for $X \rightarrow Y$ from $\pi_\emptyset(F_1)$ by using the Armstrong's axioms (since Armstrong's axioms are complete [9]). By replacing each FD in $\pi_\emptyset(F_1)$ in the proof sequence by a corresponding TFD in F_1 and replacing each Armstrong's axiom by the corresponding finite inference axiom, we know $F_1 \vdash_f X \longrightarrow_{\mu'} Y$ for some μ'. Also, by the minimality of F_1, there is no proper subset F_1' of F_1 such that $F_1' \vdash_f X \longrightarrow_{\nu'} Y$ for any ν'. It follows from Lemma 3 that $\mu' = glb(F_1)$. That is, $F_1 \vdash_f X \longrightarrow_{glb(F_1)} Y$, and hence $F \vdash_f X \longrightarrow_{glb(F_1)} Y$.

Let $F_1, \ldots, F_n$ be all the minimal supports for $X \rightarrow Y$. For each $1 \leq i \leq n$, let $\mu_i = glb(F_i)$. We know that $F \vdash_f X \longrightarrow_{\mu_i} Y$ for each $1 \leq i \leq n$. Hence, $X \longrightarrow_{\mu_i} Y \in \overline{F}^+$ for each $1 \leq i \leq n$. We have only to show that $\mu \preceq_C \{\mu_1, \ldots, \mu_n\}$.

Suppose by contradiction that this is not the case (i.e., $\mu \not\preceq_C \{\mu_1, \ldots, \mu_m\}$). From the definition of collective finer-than relation, the following holds:

$$\exists i \, \forall k \quad 1 \leq k \leq m \quad \forall j \;\; \mu(i) \not\subseteq \mu_k(j)$$

This is equivalent to saying that there exists a certain tick i of μ such that no tick of μ_k, for each

[4] For those who are not familiar with the Armstrong's axioms, the axioms are: (i) Reflexivity: $X \rightarrow Y$ if $Y \subseteq X$, (ii) Augmentation: $XW \rightarrow YW$ if $X \rightarrow Y$, and (ii) Transitivity: $X \rightarrow Z$ if $X \rightarrow Y$ and $Y \rightarrow Z$. The Armstrong's axioms are sound and complete. That is, for each set functional dependencies, a functional dependency is logically implied by this set iff it is derived by a proof sequence using the three axioms. See [9] for details.

$1 \leq k \leq m$, covers $\mu(i)$. Clearly, $\mu(i) \neq \emptyset$. Two cases arise: (1) $\mu_k = \mu_{\text{Top}}$ for some k and (2) $\mu_k \neq \mu_{\text{Top}}$ for all k. Case (1) trivially leads to a contradiction since every tick of μ is covered by the only tick of μ_{Top} (using the fact that $\mu_{\text{Top}}(0) = \mathcal{R}$). Consider case (2), i.e., $\mu_k \neq \mu_{\text{Top}}$ for all k. In this case, $F_k \neq \emptyset$ for each k since $\mu_k = glb(F_k)$. Since each μ_k is the glb of the temporal types appearing in F_k, there exists at least one TFD $V \longrightarrow_\nu W$ in F_k such that there is no tick of ν covering $\mu(i)$, i.e., there exists no j such that $\mu(i) \subseteq \nu(j)$. Then $V \rightarrow W \notin \pi_{\mu(i)}(F)$ by definition. Hence, $\pi_\emptyset(F_k) \not\subseteq \pi_{\mu(i)}(F)$. This holds for each k. On the other hand, since $F \models X \longrightarrow_\mu Y$ and $\mu(i) \neq \emptyset$, $\pi_{\mu(i)}(F) \models X \rightarrow Y$ by Lemma 2. Hence, $\pi_{\mu(i)}(F)$ is a non-temporal support for $X \rightarrow Y$. Since $F_1, \ldots, F_m$ are all the minimal non-temporal supports for $X \rightarrow Y$, there must exist $1 \leq k \leq m$ such that $\pi_\emptyset(F_k) \subseteq \pi_{\mu(i)}(F)$ This is a contradiction. Hence, we have shown that if $F \models X \longrightarrow_\mu Y$ then there exists a set $\{X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_m} Y\} \subseteq \overline{F}^+$ such that $\mu \preceq_C \{\mu_1, \ldots, \mu_m\}$, i.e., the Three Finite Axioms are complete up to descendability. $\square$

3.2 Closure of Attributes

As in the traditional relational dependency theory, we wish to give a test to verify if a TFD of the form $X \longrightarrow_\mu B$ is implied from a set of TFDs. For this purpose we introduce the (temporal) notion of finite closures of attributes and give an algorithm to compute the finite closures. First, the definition:

Definition (Finite closure of attributes)
Let F be a finite set of TFDs. For each finite set X of attributes, the *finite closure* of X wrt F is defined as

$$\overline{X}^+ = \{(B, \mu) \mid X \longrightarrow_\mu B \in \overline{F}^+ \text{ and there is no } X \longrightarrow_\nu B \text{ in } \overline{F}^+ \text{ such that } \mu \prec \nu\}$$

Note that $\overline{X}^+$ for each X is a finite set, since $\overline{F}^+$ is finite.

Proposition 2 Let F be a finite set of TFDs and X a finite set of attributes. The following holds:

$$F \models X \longrightarrow_\mu B \quad \text{iff} \quad \text{there exists } \{(B, \mu_1), \ldots, (B, \mu_m)\} \subseteq \overline{X}^+ \text{ such that } \mu \preceq_C \{\mu_1, \ldots, \mu_m\}$$

Proof. ($\Rightarrow$) Let $\mathcal{U} = \{\nu | (B, \nu) \in \overline{X}^+\}$. We only need to show that $\mu \preceq_C \mathcal{U}$. Since $F \models X \longrightarrow_\mu B$, by Theorem 1, there exists a set $\{X \longrightarrow_{\mu_1} B, \ldots, X \longrightarrow_{\mu_m} B\} \subseteq \overline{F}^+$ such that $\mu \preceq_C \{\mu_1, \ldots, \mu_m\}$. Let $\mathcal{V} = \{\nu | X \longrightarrow_\nu B \in \overline{F}^+\}$. Clearly, $\{\mu_1, \ldots, \mu_m\} \subseteq \mathcal{V}$ and $\mu \preceq_C \mathcal{V}$. By definition, $\mathcal{U} \subseteq \mathcal{V}$ and for each ν in $\mathcal{V}$, there exists ν' in $\mathcal{U}$ such that $\nu \preceq \nu'$. Since $\mu \preceq_C \mathcal{V}$, it is now clear that $\mu \preceq_C \mathcal{U}$ (note that for an arbitrary set $\mathcal{U}'$ of temporal types and temporal types ν_1 and ν_2 with $\nu_1 \preceq \nu_2$, $\mu \preceq_C \mathcal{U}' \cup \{\nu_1, \nu_2\}$ implies $\mu \preceq_C \mathcal{U}' \cup \{\nu_2\}$).

($\Leftarrow$) If $\{(B,\mu_1),\ldots,(B,\mu_m)\} \subseteq \overline{X}^+$ then, by definition of $\overline{X}^+$, $X \longrightarrow_{\mu_i} B \in \overline{F}^+$ for $1 \leq i \leq m$ and, by definition of $\overline{F}^+$, $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu_i} B$. Since the Three Finite Axioms can be derived by the Four Inference Axioms, $F \vdash X \longrightarrow_{\mu_i} B$. Applying the descendability axiom, we have $F \vdash X \longrightarrow_{\mu} B$. Thus, $F \models X \longrightarrow_{\mu} B$ by the completeness of the Four Inference Axioms. □

From the previous proposition, we know that if we have an effective procedure for obtaining the finite closure for X and an effective procedure for testing the $\preceq_C$ relation, we can then effectively decide whether a TFD $X \longrightarrow_{\mu} B$ is logically implied by F even if we may have an infinite number of logically implied TFDs. In Figure 3, we provide an algorithm for $\overline{X}^+$.

Algorithm for Computing $\overline{X}^+$

INPUT: A finite set of attributes U, a set of functional dependencies F on U, and a set $X \subseteq U$.

OUTPUT: $\overline{X}^+$, the finite closure of X with respect of F.

METHOD: We compute a sequence of sets $X^{(0)}, X^{(1)}, \ldots$ whose elements are pairs (attribute, temporal-type).

1. Let $X^{(0)} = \{(A, \mu_{\mathrm{Top}}) \mid A \in X\}$.

2. For each TFD $A_1 \ldots A_k \longrightarrow_{\mu} B_1 \ldots B_m$ in F such that $\{(A_1,\mu_1),\ldots,(A_k,\mu_k)\}$ is a subset of $X^{(i)}$ we compute the set $\{(B_l,\mu') \mid 1 \leq l \leq m, \quad \mu' = glb(\mu_1,\ldots,\mu_k,\mu)\}$. Let $f_1,\ldots,f_r$ be all the TFDs in F that satisfy the above condition and $Y_1,\ldots,Y_r$ the corresponding computed sets. Then let $X^{(i+1)} = X^{(i)} \cup Y_1 \cup \ldots \cup Y_r$.

Step 2. is repeated until $X^{(i+1)} = X^{(i)}$. The Algorithm returns the set

$$X^{(i)} \setminus \{(B,\mu') \in X^{(i)} \mid \exists \nu \ (B,\nu) \in X^{(i)} \text{ with } \mu' \prec \nu\}$$

Figure 3: Algorithm for computing $\overline{X}^+$.

Theorem 3 The algorithm in Figure 3 correctly computes $\overline{X}^+$ in a finite number of steps.

See Appendix A.1 for the proof.

Example 3 As an example, let $\mathtt{w_r}$ be the temporal type of "recent weeks" defined as follows: $\mathtt{w_r}(1)$ maps to the week starting July 4, 1994, and $\mathtt{w_r}(2)$ to the week after that, and so on. Now let $F =$

$\{A \longrightarrow_{\mathtt{wr}} B, B \longrightarrow_{\mathtt{month}} A\}$. It is easily seen that $\overline{A}^+ = \{(A, \text{Top}), (B, \mathtt{wr})\}$, $\overline{B}^+ = \{(B, \text{Top}), (A, \mathtt{month})\}$ and $\overline{AB}^+ = \overline{A}^+ \cup \overline{B}^+$. $\square$

4 Temporal Normalization

In this section, we extend the traditional normalization theory to temporal databases. Thus, TFDs not only capture certain type of constraints in temporal databases, but can also be used to eliminate data redundancy and other anomalies in temporal relations. We begin by reconsidering the temporal module (ACCOUNTS, second, ϕ) given in Example 1.

Example 4 As discussed in the introduction, the temporal module scheme (ACCOUNTS, second) given in Example 1 should be decomposed into two temporal module schemes (TRANSACTION-INFO, second) and (ACCUM-INTEREST, day) where

$$\mathtt{TRANSACTION\text{-}INFO} = \langle \mathtt{AcctNo,\ Amount,\ Balance} \rangle$$

and ACCUM-INTEREST $= \langle$AcctNo, AccumInt$\rangle$. The time windowing function ϕ given earlier will be also "decomposed" into two time windowing functions ϕ_1 and ϕ_2, defined as follows:

$$\begin{array}{ll}
\phi_1(3/3/93:09:01:00) = \{\langle 1001, +1000, 1000\rangle\} & \phi_2(3/3/93) = \{\langle 1001, 0.00\rangle\} \\
\phi_1(3/4/93:10:01:55) = \{\langle 1001, -500, 500\rangle\} & \phi_2(3/4/93) = \{\langle 1001, 0.14\rangle\} \\
\phi_1(3/4/93:11:00:00) = \{\langle 1001, +200, 700\rangle\} & \\
\phi_1(3/4/93:12:19:03) = \{\langle 1001, -315, 385\rangle\} & \\
\phi_1(3/4/93:18:00:00) = \{\langle 1001, -255, 130\rangle\} & \\
\phi_1(3/7/93:09:00:00) = \{\langle 1001, -10, 120\rangle\} & \phi_2(3/7/93) = \{\langle 1001, 0.19\rangle\} \\
\phi_1(3/7/93:12:01:40) = \{\langle 1001, +100, 220\rangle\} &
\end{array}$$

$\square$

In order to have meaningful decompositions of temporal modules, we need to define how we can join decomposed temporal modules to recover original temporal modules. To this end, we define temporal natural join and temporal projection operations.

Definition Let $\mathtt{M}_1 = (R_1, \mu, \phi_1)$ and $\mathtt{M}_2 = (R_2, \mu, \phi_2)$ be two temporal modules in terms of the *same* temporal type μ. Then $\mathtt{M}_1 \bowtie_T \mathtt{M}_2$, called *temporal natural join* of $\mathtt{M}_1$ and $\mathtt{M}_2$, is the temporal module $\mathtt{M} = (R_1 \cup R_2, \mu, \phi)$, where ϕ is defined as follows: For each $i \geq 1$, $\phi(i) = \phi_1(i) \bowtie \phi_2(i)$, where $\bowtie$ is the traditional natural join operation (cf. [9]).

Thus, temporal natural join of two temporal modules is obtained by taking the natural joins of their snapshots. Temporal projection of temporal modules is defined similarly. Basically, the projection of a temporal module is a collection of snapshot projections.

Definition Let $\mathtt{M} = (R, \mu, \phi)$ and $R_1 \subseteq R$. Then $\pi^T_{R_1}(\mathtt{M})$, called *the projection of* M *on* R_1, is the temporal module (R_1, μ, ϕ_1), where ϕ_1 is defined as follows: For each $i \geq 0$, $\phi_1(i) = \pi_{R_1}(\phi(i))$, where π is the traditional projection operation (cf. [9]).

We define the projection of a set of TFDs analogously to the standard definition of projection of a set of functional dependencies.

Definition Given a set of TFDs F, the *projection of* F *onto a set of attributes* Z, denoted $\pi_Z(F)$, is the set of TFDs $X \longrightarrow_\nu Y$ logically implied by F such that $XY \subseteq Z$.

Note that the projection only takes into account the attributes, not the underlying temporal types.

As we have seen for F, $\pi_Z(F)$ can also be infinite. However, since we know how to compute a finite cover of the closure of attributes, we can compute a finite cover of the projection of F on Z as follows: Let

$$\overline{\pi}_Z(F) = \{X \longrightarrow_\nu A_1 \cdots A_m \mid XA_1 \cdots A_m \subseteq Z \text{ and } (A_i, \nu) \in \overline{X}^+ \text{ for } 1 \leq i \leq m\}.$$

Clearly, $\overline{\pi}_Z(F)$ is a finite set. By the completeness of the Three Finite Inference Axioms and the definition of $\overline{F}^+$, we can easily see that $\overline{\pi}_Z(F)$ is a "finite cover" of $\pi_Z(F)$. That is:

Proposition 3 The following holds:

$$\pi_Z(F) = \{X \longrightarrow_{\nu'} A_1 \cdots A_m \mid X \longrightarrow_{\nu_i} A_i \in \overline{\pi}_Z(F) \text{ for } 1 \leq i \leq m \text{ and } \nu' \preceq_C \{\nu_1, \ldots, \nu_m\}\}.$$

Before we define lossless decompositions, we need first to introduce three auxiliary operations, **Down**, **Up** and $\cup_{\mathrm{T}}$, on temporal modules.

Definition For $\mathtt{M} = (R, \mu, \phi)$ and temporal types ν_1 and ν_2, let $\mathbf{Down}(\mathtt{M}, \nu_1)$ and $\mathbf{Up}(\mathtt{M}, \nu_2)$ be the temporal modules (R, ν_1, ϕ_1) and (R, ν_2, ϕ_2), respectively, where ϕ_1 and ϕ_2 are defined as follows: For each $i \geq 1$, let

$$\phi_1(i) = \begin{cases} \emptyset & \text{if } \nu_1(i) = \emptyset \\ \emptyset & \text{if there is no } j \text{ such that } \nu_1(i) \subseteq \mu(j) \\ \phi(j) \text{ where } \nu_1(i) \subseteq \mu(j) & \text{otherwise} \end{cases}$$

and

$$\phi_2(i) = \bigcup_{j:\mu(j)\subseteq\nu_2(i)} \phi(j).$$

Note that if there is no j such that $\mu(j) \subseteq \nu_2(i)$, then $\phi_2(i) = \emptyset$.

Intuitively, function **Down** maps a temporal module in terms of temporal type μ to a temporal module in terms of a finer temporal type such that each tuple that is valid at tick i in μ is taken to be valid at all ticks j in ν_1 provided $\nu_1(j) \subseteq \mu(i)$. For example, consider a temporal module $\mathtt{M}$ that registers faculty ranks in terms of the temporal type `academic year`. Then $\mathbf{Down}(\mathtt{M}, \mathtt{month})$ converts $\mathtt{M}$ into a temporal module in terms of `month` with a windowing function that gives for each month the rank of the faculty member during the corresponding academic year. Similarly, the function **Up** is used to obtain a temporal module in terms of temporal type ν_2 from a temporal module that is in terms of a finer temporal type μ by taking each fact that is valid in a tick i in terms of μ to be valid in tick j in terms of ν_2 provided $\mu(i) \subseteq \nu_2(j)$. Consider a temporal module $\mathtt{M}$ that registers faculty ranks in terms of the temporal type `month`. Then $\mathbf{Up}(\mathtt{M}, \mathtt{academic\text{-}year})$ is a temporal module that registers faculty ranks in terms of academic year, with a windowing function that gives for an academic year all the rank(s) of faculty members during the months of the academic year.

Definition Given temporal modules $\mathtt{M}_1, \ldots, \mathtt{M}_n$ over the same relation R and temporal type μ, their union is defined as the new module $\mathtt{M} = (R, \mu, \phi)$, where ϕ, for each tick, is simply the union of the values of the windowing functions for the same tick in the other modules. Formally, if $\mathtt{M}_j = (R, \mu, \phi_j)$ for $j = 1, \ldots, n$, then $\mathtt{M}_1 \cup_{\mathrm{T}} \ldots \cup_{\mathrm{T}} \mathtt{M}_n = (R, \mu, \phi)$ where, for each $i \geq 1$ $\phi(i) = \bigcup_{1\leq j\leq n} \phi_j(i)$.

Definition We say that $(R, \mu_1), \ldots, (R, \mu_n)$ is a *lossless union decomposition* of (R, μ) if for each module $\mathtt{M}$ on (R, μ), $\mathtt{M} = \mathbf{Up}(\mathbf{Down}(\mathtt{M}, \mu_1), \mu) \cup_{\mathrm{T}} \ldots \cup_{\mathrm{T}} \mathbf{Up}(\mathbf{Down}(\mathtt{M}, \mu_n), \mu))$.

In particular, we observe that if $\mu_1, \ldots, \mu_n$ are types such that their non-empty ticks form a partition of the non-empty ticks of μ, the above condition is satisfied. This intuitively says that we can always decompose a scheme so that the union of the ticks in the different temporal types is the original type and there is no intersection in the sets of ticks. As an example, if we have a scheme in days, we can decompose it in two schemes, such that in the type of the first we just consider days from Monday through Friday and in the other we consider Saturdays and Sundays. Our modules can be represented in the original scheme or in the two new schemes without losing information. Notice, however, that this decomposition may not preserve temporal functional dependencies. Indeed, consider the TFD $A \longrightarrow_{\mathtt{week}} B$, where A and B are the attributes of the module scheme, and assume we store information by the two schemes as above. Suppose the facts (a,b) and (a,c) hold on Monday and Sunday of a particular week, and there are no other facts in the module. Clearly, the two modules satisfy the TFDs separately, but not collectively. We will study dependency-preserving decompositions in Section 6.

We now define the general notion of a lossless decomposition of a temporal module using the three auxiliary operations **Down**, **Up** and $\cup_{\mathrm{T}}$.

Definition (Lossless decomposition)
Let (R,μ) be a temporal module scheme and F a set of TFDs. A finite set ρ of temporal module schemes is said to be a *lossless decomposition of* (R,μ) *wrt* F if there exist subsets $\rho_1,\ldots,\rho_m$ of ρ such that for each temporal module $\mathtt{M}$ on (R,μ) that satisfies all TFDs in F, we have

$$\mathtt{M} = \mathbf{Join}(\rho_1) \cup_{\mathrm{T}} \cdots \cup_{\mathrm{T}} \mathbf{Join}(\rho_m),$$

where

$$\mathbf{Join}(\rho_i) = \mathbf{Down}(\mathbf{Up}(\pi^T_{R^i_1}(\mathtt{M}),\mu^i_1),\mu) \bowtie_T \cdots \bowtie_T \mathbf{Down}(\mathbf{Up}(\pi^T_{R^i_k}(\mathtt{M}),\mu^i_k),\mu)$$

for each $\rho_i = \{(R^i_1,\mu^i_1),\ldots,(R^i_k,\mu^i_k)\}$.

Example 5 Now we are in a position to describe how we can recover the `ACCOUNTS` temporal module from `TRANSACTION-INFO` and `ACCUM-INTEREST` temporal modules. Since `ACCOUNTS` is in terms of temporal type `second`, we take the temporal join of `TRANSACTION-INFO` and **Down**(`ACCUM-INTEREST`, `second`) to recover `ACCOUNTS`. □

We introduce an equivalent notion of lossless decomposition for technical reasons. Suppose ρ is a lossless decomposition of (R,μ) and $\mathtt{M} = (R,\mu,\phi)$ is a temporal module on (R,μ). Then for each tick i of μ, the relation $\phi(i)$ can always be recovered from the projections of $\mathtt{M}$ over the decomposition. This leads to the notion of tickwise lossless decomposition. For simplicity of presentation, we introduce the symbol **MaxSub**; we use it as a function such that for each tick i of μ and a set ρ of schemes, $\mathbf{MaxSub}(\mu(i),\rho)$ is the maximal subset of ρ such that for each temporal type associated with these schemes there exists a tick covering tick i of μ. That is,

$$\mathbf{MaxSub}(\mu(i),\rho) = \{(R,\nu) \in \rho \mid \mu(i) \subseteq \nu(j) \text{ for some j}\}.$$

This allows us to consider only that part of the decomposition which contributes to the recovery of the original module at tick i of μ.

Definition (Tickwise lossless decomposition)
Let ρ be a decomposition of schema (R,μ) and F a set of TFDs. The decomposition ρ is said to

be a *tickwise lossless decomposition of* (R, μ) *wrt* F if for each nonempty tick k of μ, the following holds: If $\mathbf{MaxSub}(\mu(k), \rho) = \{(R_1, \mu_1), \ldots, (R_m, \mu_m)\}$ and, for each $1 \leq i \leq m$, $(R_i, \mu, \phi_i) = \mathbf{Down}(\mathbf{Up}(\pi^T_{R_i}(\mathtt{M}), \mu_i), \mu)$ and k_i is the integer such that $\mu(k) \subseteq \mu_i(k_i)$, then $\phi(k) = \phi_1(k_1) \bowtie \ldots \bowtie \phi_m(k_m)$.

The above definition intuitively says that for each tick of μ we can reconstruct the original module from its decomposition.

Proposition 4 Let (R, μ) a temporal module scheme, F a set of TFDs, and ρ a decomposition of (R, μ). Then ρ is a lossless decomposition of (R, μ) wrt F iff it is a tickwise lossless decomposition of (R, μ) wrt F.

Proof. First we establish the "if" ($\Leftarrow$) part. Let $P = \{\mathbf{MaxSub}(\mu(k), \rho) : \mu(k) \neq \emptyset\}$. Since $\mathbf{MaxSub}(\mu(k), \rho)$ is a subset of ρ for each non-empty tick k of μ and ρ is a finite set, it follows that that P is also a finite set. Assume $P = \{\rho_1, \ldots, \rho_m\}$. Now let $\mathtt{M} = (R, \mu, \phi)$ be a temporal module that satisfies F and, for each $1 \leq i \leq m$, let

$$\mathtt{M}_i = \mathbf{Down}(\mathbf{Up}(\pi^T_{R^i_1}(\mathtt{M}), \mu^i_1), \mu) \bowtie_T \cdots \bowtie_T \mathbf{Down}(\mathbf{Up}(\pi^T_{R^i_n}(\mathtt{M}), \mu^i_n), \mu),$$

assuming $\rho_i = \{(R^i_1, \mu^i_1), \ldots, (R^i_n, \mu^i_n)\}$. Let $1 \leq i \leq m$. Suppose $\mathtt{M}_i = (R_i, \mu_i, \phi_i)$. It follows from the definition of $\mathtt{M}_i$ that $\nu_i = \mu$. Furthermore, since ρ is tickwise lossless wrt F, we know that $R_i = R$ and $\phi_i(k) = \phi(k)$ for each non-empty tick k with $\mathbf{MaxSub}(\mu(k), \rho) = \rho_i$. Let $\mathtt{M}' = (R, \mu, \phi') = \mathtt{M}_1 \cup_{\mathrm{T}} \cdots \cup_{\mathrm{T}} \mathtt{M}_m$. It thus follows that $\phi(k) \subseteq \phi'(k)$ for each non-empty tick k of μ. Let k be a non-empty tick of μ. We show that $\phi'(k) \subseteq \phi(k)$. To do this, assume $\phi'(k) \not\subseteq \phi(k)$. Thus, there exists a tuple t such that t is in $\phi'(k)$ but not in $\phi(k)$. Without loss of generality, suppose $\mathbf{MaxSub}(\mu(k), \rho) = \rho_1$ and we know $\phi(k) = \phi_1(k)$ by the fact that ρ is tickwise lossless. Hence, t is not in $\phi_1(k)$ since t is not in $\phi(k)$. Let $\mathtt{M}'' = (R, \mu, \phi'') = \mathtt{M}_2 \cup_{\mathrm{T}} \cdots \cup_{\mathrm{T}} \mathtt{M}_m$. Clearly, t must be in $\phi''(k)$ by the definition of $\cup_{\mathrm{T}}$. Hence, there must exist j with $2 \leq j \leq m$ such that t is in $\phi_j(k)$. This is impossible. Indeed, since $\rho_j \neq \rho_1$, there exists (S, ν) in ρ_j such that $\mu(k) \not\subseteq \nu(l)$ for all l. Let $\mathtt{M}_s = (S, \mu, \phi_s) = \mathbf{Down}(\mathbf{Up}(\pi^T_S(\mathtt{M}), \nu), \mu)$. Since $\mu(k) \not\subseteq \nu(l)$ for all l, we know $\phi_s(k) = \emptyset$ by the definition of $\mathbf{Down}$. Thus, $\phi_j(k) = \emptyset$ because $\phi_j(k)$ is the result of joining $\phi_s(k)$ with other relations and $\phi_s(k) = \emptyset$. Therefore, t cannot be in $\phi_j(k)$ for each $2 \leq j \leq m$. This is a contradiction and we conclude that $\phi(k) = \phi'(k)$. Hence, $\phi = \phi'$ and $\mathtt{M}' = \mathtt{M}$. This shows that ρ is a lossless decomposition of (R, μ) wrt F.

We now establish the "only-if" ($\Rightarrow$) part. Assume that ρ is a lossless decomposition of (R, μ). Then there exist subsets $\rho_1, \ldots, \rho_m$ of ρ having the condition given in the definition. Let $\mathtt{M} = (R, \mu, \phi)$ be a temporal module that satisfies F and, for each $1 \leq i \leq m$, let

$$\mathtt{M}_i = \mathbf{Down}(\mathbf{Up}(\pi^T_{R^i_1}(\mathtt{M}), \mu^i_1), \mu) \bowtie_T \cdots \bowtie_T \mathbf{Down}(\mathbf{Up}(\pi^T_{R^i_n}(\mathtt{M}), \mu^i_n), \mu),$$

assuming $\rho_i = \{(R^i_1, \mu^i_1), \ldots, (R^i_n, \mu^i_n)\}$. Let $1 \leq i \leq m$. Suppose $\mathtt{M}_i = (R_i, \mu_i, \phi_i)$. It follows from the definition of $\mathtt{M}_i$ that $\mu_i = \mu$. Furthermore, since $\rho_1, \ldots, \rho_m$ satisfy the condition given in the definition of lossless decomposition, it is easily seen that $R_i = R$. Let k be a non-empty tick of μ. It is also easily seen that, for each $1 \leq j \leq n$, if $\mu(k) \not\subseteq \mu^i_j(l_j)$ for all l_j, then $\phi_i(k) = \emptyset$ by the definition of the **Down** operation. Thus, if $\phi_i(k) \neq \emptyset$, then $\mu(k) \subseteq \mu^i_j(l_j)$ for some l_j and hence, $\rho_i \subseteq \mathbf{MaxSub}(\mu(k), \rho)$. Assume that $\rho_i = \{(R^i_1, \mu^i_1), \ldots, (R^i_p, \mu^i_p)\}$, and let $(R^i_j, \mu, \phi^i_j) = \mathbf{Down}(\mathbf{Up}(\pi_{R^i_j}(\mathtt{M}), \mu^i_j), \mu)$ for each $1 \leq j \leq p$. Since $\rho_i \subseteq \mathbf{MaxSub}(\mu(k), \rho)$, for each $1 \leq j \leq p$, there exists l_j such $\mu(k) \subseteq \mu^i_j(l_j)$. By definition of **Up** and **Down**, $\pi_{R^i_j}(\phi(k)) \subseteq \phi^i_j(k)$ for each $1 \leq j \leq p$. Therefore, $\phi(k) \subseteq \phi^i_1(k) \bowtie \cdots \bowtie \phi^i_p(k)$. Since $\phi_i(k) = \phi^i_1(k) \bowtie \cdots \bowtie \phi^i_p(k)$, we have $\phi(k) \subseteq \phi_i(k)$. Furthermore, since $\phi(k) = \phi_1(k) \cup \cdots \cup \phi_m(k)$ by the lossless property of the decomposition ρ, it is easily seen that $\phi_i(k) \subseteq \phi(k)$. Hence, we have $\phi_i(k) = \phi(k)$. Suppose now that a scheme (R_j, μ_j) in $\rho - \rho_i$ and (R_j, μ_j) is in $\mathbf{MaxSub}(\mu(k), \rho)$. Since (R_j, μ_j) is in $\mathbf{MaxSub}(\mu(k), \rho)$, there exists q such that $\mu(i) \subseteq \mu_j(q)$. Hence, by the definition of **Up** and **Down**, $\pi_{R_j}(\phi(k)) \subseteq \phi_j(k)$. It is easily seen that $\phi(k) = \phi(k) \bowtie \phi_j(k)$. Therefore, $\phi(k) = \phi_i(k) \bowtie \phi_j(k)$. That is, given a scheme in $\mathbf{MaxSub}(\mu(k), \rho)$ that is not in ρ_i, the join of $\phi_i(k)$ with the windowing function obtain from this scheme will keep the value of $\phi(k)$. Therefore, we conclude that the decomposition ρ is tickwise lossless. □

The following theorem gives a sufficient condition for a lossless decomposition.

Theorem 4 Let $\mathbf{F}$ be a set of TFDs. The decomposition (R_1, μ) and (R_2, ν) of $(R_1 \cup R_2, \mu)$, where $\mu \preceq \nu$, is lossless wrt F if for all i_1 and i_2, $\mu(i_1, i_2) \subseteq \nu(j)$ for some j implies that $R_1 \cap R_2 \rightarrow R_2$ is in $\pi_{\mu(i_1,i_2)}(F)$.

Proof. Let $\mathtt{M} = (R_1 \cup R_2, \mu, \phi)$ be a temporal module that satisfies all TFDs in F. Let $(R_1, \mu, \phi_1) = \pi^T_{R_1}(\mathtt{M})$ and $(R_2, \nu, \phi_2) = \mathbf{Up}(\pi^T_{R_2}(\mathtt{M}), \nu)$. We only need to show that for each positive integer k, $\phi(k) = \phi_1(k) \bowtie \phi_2(k')$, where $\mu(k) \subseteq \nu(k')$, since this implies that the decomposition is tickwise lossless and hence lossless by Proposition 4. Let k be a positive integer. If $\mu(k) = \emptyset$, we have $\phi(k) = \emptyset$ and $\phi_1(k) = \emptyset$. Therefore, $\phi(k) = \phi_1(k) \bowtie \phi_2(k')$ holds. Now suppose $\mu(k) \neq \emptyset$. Since $\mu \preceq \nu$, there exists a positive integer k' such that $\mu(k) \subseteq \nu(k')$. It is easily seen that $\phi(k) \subseteq \phi_1(k) \bowtie \phi_2(k')$ by the definitions of temporal projection, **Up** and the natural join. Now assume that a tuple t is in $\phi_1(k) \bowtie \phi_2(k')$. We only need to show that t is in $\phi(k)$. Since t is in $\phi_1(k) \bowtie \phi_2(k')$, there exists t_1 in $\phi_1(k)$ and t_2 in $\phi_2(k')$

such that $t[R_1] = t_1$ and $t[R_2] = t_2$. By definition, there exists t'_1 in $\phi(k)$ such that $t'_1[R_1] = t_1$ and there exists k'' and t'_2 in $\phi(k'')$ such that $\mu(k, k'') \subseteq \nu(k')$ and $t'_2[R_2] = t_2$. Since $\mu(k, k'') \subseteq \nu(k')$, by hypothesis, we know that $R_1 \cap R_2 \rightarrow R_2$ is in $\pi_{\mu(k,k'')}(F)$. Since $R_1 \cap R_2 \subseteq R_1$ and $R_1 \cap R_2 \subseteq R_2$, we have $t'_1[R_1 \cap R_2] = t_1[R_1 \cap R_2] = t[R_1 \cap R_2] = t_2[R_1 \cap R_2] = t'_2[R_1 \cap R_2]$. It follows from (1) $t \in \phi(k)$ and $t'_2 \in \phi(k'')$ and (2) $R_1 \cap R_2 \rightarrow R_2 \in \pi_{\mu(k,k'')}(F)$ that $t'_1[R_2] = t'_2[R_2]$. Combining this with the fact $t[R_1] = t_1 = t'_1[R_1]$, we know $t[R_1 \cup R_2] = t'_1[R_1 \cup R_2]$, i.e., $t = t'_1$. Thus, t is in $\phi(k)$ as desired. □

From this theorem, it follows that the decomposition given in Example 1 is lossless.

5 Temporal BCNF

In this section, we define the temporal analog of Boyce-Codd Normal Form (BCNF). The temporal BCNF (TBCNF) retains the spirit of BCNF. That is, TBCNF does not allow any redundancy introduced by TFDs. We then give a decomposition algorithm that renders lossless TBCNF decomposition for any given temporal module scheme.

In order to define TBCNF, we need the notion of keys.

Definition Let F be a set of TFDs defined on a temporal module scheme (R, μ). A set of attributes $X \subseteq R$ is said to be a *temporal superkey of* (R, μ) if $X \longrightarrow_\mu R$ is logically implied by the TFDs in F.[5]

If `M` is a temporal module on (R, μ) and X is a superkey of (R, μ), then whenever t_1 and t_2 are two tuples with $t_1[X] = t_2[X]$ for the same tick in μ, then $t_1 = t_2$.

A temporal superkey X of (R, μ) is called a *temporal candidate key* if for each A in X, $X - \{A\}$ is not a temporal superkey of (R, μ).

We are now ready to define the concept of temporal BCNF:

Definition (Temporal BCNF)
A temporal module scheme (R, μ), with a set F of TFDs, is said to be in *temporal BCNF* (TBCNF) if for each TFD $X \longrightarrow_\nu Y$ that is logically implied by F, where (a) $XY \subseteq R$, (b) $Y \not\subseteq X$, and (c) at least one tick of μ is covered by some tick of ν, the following two conditions hold:

(i) $X \longrightarrow_\mu R$ is logically implied by F, i.e., X is a superkey of (R, μ), and

(ii) For all non-empty ticks i_1 and i_2 of μ, with $i_1 \neq i_2$, $X \rightarrow Y \notin \pi_{\mu(i_1,i_2)}(F)$.

[5]This notion of temporal superkey in terms of snapshots is closely related to the "proper temporal database" notion in [13], and is identical to the concept of keys in [3].

While the first condition is the temporal analog of the traditional definition of BCNF, the second condition disallows redundancy due to temporal functional dependencies. Indeed, we have redundancy whenever there exists a TFD with a temporal type ν such that two ticks of the temporal type of the module are covered by the same tick of ν. In this case if we have two tuples separately on these two ticks, one of them may have redundant information. Thus, the two conditions for the TBCNF eliminate all possible data redundancy that may arise by the presence of TFDs.

Example 6 The temporal module scheme (`ACCOUNTS`, `second`) in Example 1 is not in TBCNF since $F \models \texttt{AcctNo} \longrightarrow_{\texttt{day}} \texttt{AccumInt}$. However, both (`TRANSACTION-INFO`, `second`) and (`ACCUM-INTEREST`, `day`) are in TBCNF. □

As in the traditional relational theory, decomposition is used to convert temporal module schemes that are not in TBCNF into schemes that are in TBCNF. Similar to the traditional BCNF, the price we pay for such a decomposition is that some of the TFDs may not be preserved. We will discuss dependency-preserving decompositions in Section 6.

5.1 Decomposing temporal module schemes into TBCNF

In this section, we describe an algorithm that decomposes temporal schemes into TBCNF.

In the algorithm we will use two operators to create a new temporal type from a given one. The first one is called the *collapsing* operator which, given a temporal type μ and a positive integer i, gives a type μ_c by combining tick i and $i+1$ of μ into one tick and retaining all other ticks of μ. Formally, μ_c is the temporal type defined as follows: For all $j \geq 1$, let

$$\mu_c(j) = \begin{cases} \mu(j) & \text{for } 1 \leq j \leq i-1 \\ \mu(i, i+1) & \text{for } j = i \\ \mu(j+1) & \text{for } j > i \end{cases}$$

The second operator is called the *pruning* operator which, given a temporal type μ and a positive integer i, produces a type μ_d by dropping the tick i of μ and keeping all other ticks of μ. Formally, μ_d is the temporal type defined as follows: For all $j \geq 1$, let

$$\mu_d(j) = \begin{cases} \mu(j) & \text{for } 1 \leq j \leq i-1 \\ \mu(j+1) & \text{for } j \geq i \end{cases}$$

Figure 4 presents the TBCNF decomposition algorithm. We note that if a scheme (R_i, μ_i) is not in TBCNF wrt $\pi_{R_i}(F)$, then there exists a TFD in $\overline{\pi}_{R_i}(F)$ such that one of the two conditions in the

Algorithm for TBCNF Decomposition	
INPUT:	A temporal module scheme (R, μ) and a set F of temporal functional dependencies.
OUTPUT:	A lossless decomposition ρ of (R, μ) such that each scheme in ρ is in TBCNF.
METHOD:	We compute a sequence $\rho^{(0)}, \rho^{(1)}, \ldots$ of decompositions of (R, μ):

Step 1. Let $\rho^{(0)} = \{(R, \mu)\}$.

Step 2. Suppose $\rho^{(j)}$ is the last set we have computed. If at least one scheme in $\rho^{(j)}$ is not in TBCNF we compute $\rho^{(j+1)}$ as follows. Take a scheme (R_i, μ_i) from $\rho^{(j)}$ that is not in TBCNF and a TFD $X \longrightarrow_{\nu} A$ from $\overline{\pi}_{R_i}(F)$ that violates the TBCNF conditions for this scheme. Let

$$\rho^{(j+1)} = (\rho^{(j)} \setminus \{(R_i, \mu_i)\}) \cup \{(R_i, \nu_1), (R_i - A, \nu_2), (XA, \nu_3)\},$$

where ν_1, ν_2, and ν_3 are new temporal types defined as follows:

- ν_1 is obtained from μ_i by recursively applying the *pruning* operator to drop all the nonempty ticks of μ_i that are covered by some tick of ν. If ν_1 results in an empty type, then the corresponding scheme, i.e., (R_i, ν_1), is not added into $\rho^{(j+1)}$.

- ν_2 is the complementary of ν_1, that is, its ticks are all the ticks of μ_i that are covered by some tick of ν.

- ν_3 is obtained from ν_2 by recursively applying the *collapsing* operator to collapse each pair of ticks of ν_2 that are covered by some tick of ν. That is, each non-empty tick of ν_3 is a combination of one or more ticks of ν_2. Moreover, no two ticks of ν_3 are covered by a single tick of ν.

Step 2 is repeated until each scheme in $\rho^{(j)}$ is in TBCNF. Then the algorithm returns $\rho^{(j)}$.

Figure 4: Algorithm for TBCNF decomposition.

definition of TBCNF is violated. Thus, the decomposition required at step 2 can always be carried out. Indeed, since (R_i, μ_i) is not in TBCNF wrt $\pi_{R_i}(F)$, there exists $X \longrightarrow_{\nu} A$ in $\pi_{R_i}(F)$ such that one of the two conditions for TBCNF is violated. By Proposition 3, there exists $\nu_1, \ldots, \nu_n$ such that $\nu \preceq_C \{\nu_1, \ldots, \nu_n\}$ and $X \longrightarrow_{\nu_j} A$ is in $\overline{\pi}_{R_i}(F)$ for each $1 \le j \le n$. If X is not a superkey of (R_i, μ_i) (i.e., the first condition for TBCNF is violated), then clearly each $X \longrightarrow_{\nu_j} A$ violates the first condition for TBCNF. Now suppose there exist integers k_1 and k_2, with $k_1 \neq k_2$, and k such that $\mu_i(k_1, k_2) \subseteq \nu(k)$ (i.e., the second condition for TBCNF is violated). Since $\nu \preceq_C \{\nu_1, \ldots, \nu_n\}$, there exist j and k' such that $\nu(k) \subseteq \nu_j(k')$. Hence, $\mu_i(k_1, k_2) \subseteq \nu(k) \subseteq \nu_j(k')$, i.e., $X \longrightarrow_{\nu_j} A$ violates the second condition for TBCNF. We conclude that if (R_i, μ_i) is not in TBCNF wrt $\pi_{R_i}(F)$, we can always find a TFD in $\overline{\pi}_{R_i}(F)$ that violates the condition for TBCNF.

Theorem 5 The algorithm in Figure 4 always terminates and gives a lossless TBCNF decomposition of the input temporal scheme wrt the input set of TFDs.

See Appendix A.2 for the proof.

Example 7 We illustrate the algorithm by applying it to the temporal module scheme (`ACCOUNTS`, `second`) of Example 1 wrt the following functional dependencies:

$$\texttt{AcctNo} \longrightarrow_{\texttt{second}} \texttt{Amount Balance AccumInt}$$

and

$$\texttt{AcctNo} \longrightarrow_{\texttt{day}} \texttt{AccumInt}.$$

It is clear that (`ACCOUNTS`, `second`) is not in TBCNF since we have `AcctNo` $\longrightarrow_{\texttt{day}}$ `AccumInt`. The second step of the algorithm determines three new temporal types: ν_1, ν_2 and ν_3. It is easily seen that ν_1 will be μ_{Bottom}, ν_2 will be `second` and ν_3 will be `day`. Since ν_1 is μ_{Bottom}, (R_i, ν_1) is not added into the decomposition. Therefore, two new schemes are used to replace the original scheme:

$$((\texttt{ACCOUNTS, Amount, Balance}), \texttt{second})$$

and

$$((\texttt{ACCOUNTS, AccumInt}), \texttt{day}).$$

Both of these are in TBCNF. □

As a final note on the TBCNF decomposition algorithm, if all TFDs are in terms of the temporal type of the given module scheme, the algorithm in Figure 4 reduces to the decomposition algorithm of traditional (non-temporal) BCNF.

6 Preservation of Dependencies

It is well-known that in order to eliminate all data redundancy due to (non-temporal) functional dependencies, we have to pay the price of losing some dependencies [9]. Since (non-temporal) BCNF is a special case of TBCNF, it is no surprise that TBCNF decomposition may not preserve all TFDs. It is a surprise that even if a module scheme has only two attributes, redundancy may not be totally eliminated without loosing TFDs.[6] As an example, consider the temporal scheme $(AB, \texttt{day})$ with TFDs $A \longrightarrow_{\texttt{week}} B$ and $A \longrightarrow_{\texttt{month}} B$. Clearly, $(AB, \texttt{day})$ is not in TBCNF. By using the TBCNF decomposition algorithm, $\{(A, \texttt{day})$ and $(AB, \texttt{month})\}$ can be the lossless TBCNF decomposition. However, the TFD $A \longrightarrow_{\texttt{week}} B$ cannot be enforced in either schemes. In fact, as we show below, there is no way of enforcing both TFDs without any data redundancy.

In order to formally capture the intuition of "enforcing TFDs", as in the traditional relational design theory, we define the notion of dependency preservation:

Definition (Dependency preservation)
Given a module scheme (R, μ), a set F of TFDs, we say that a decomposition $\rho = \{(R_1, \mu_1), \ldots, (R_k, \mu_k)\}$ preserves the dependencies in F if, for each module $\texttt{M}$ on (R, μ),

$$\mathbf{Up}(\pi^T_{R_i}(\texttt{M}), \mu_i) \text{ satisfies } \pi_{R_i}(F) \text{ for each } i = 1, 2, \ldots, k \quad \text{implies} \quad \texttt{M} \text{ satisfies } F.$$

First, we deal with the simple case where each TFD has the same left and right-hand sides (but in terms of a different temporal type). The example given in the beginning of this section is such a case. We show how these temporal functional dependencies can be preserved.

Given a temporal type μ and a set of TFDs $F = \{X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_n} Y\}$, let the function *cop*$(\mu, F)$ return a new type λ obtained starting with $\lambda = \mu$ and recursively collapsing each pair of ticks i_1 and i_2 such that both of the following conditions are satisfied:

- $X \rightarrow Y \in \pi_{\lambda(i_1,i_2)}(F)$, and
- for all i_3, $X \rightarrow Y \in \pi_{\lambda(i_1,i_3)}(F) \cup \pi_{\lambda(i_2,i_3)}(F)$ implies $X \rightarrow Y \in \pi_{\lambda(i_1,i_2,i_3)}(F)$.

The *cop* function gives a temporal type that is the coarsest of all the temporal types ν such that each tick of ν is either a tick of μ or a composition of a set of ticks of μ covered by the same TFD in F, and such that $\{(R, \nu)\}$ still preserves all TFDs in F.

[6] Note that if a (non-temporal) relation scheme has only two attributes, then the scheme is always in BCNF, and therefore has no redundancy at all.

For example, given the temporal type day and $F = \{A \longrightarrow_{\texttt{week}} B, A \longrightarrow_{\texttt{month}} B\}$, Figure 5 shows the temporal type $cop(\texttt{day}, F)$. In the figure, each interval of a temporal type corresponds to a tick of that type.

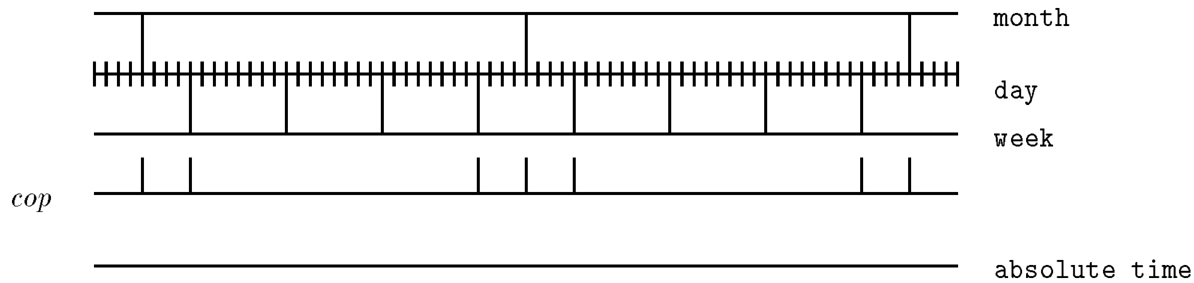

Figure 5: The *cop* function (types not to scale).

We now show that this *cop* function indeed has the required properties.

Proposition 5 Let (R, μ) be a module scheme and $F = \{X \longrightarrow_{\mu_1} Y, \ldots, X \longrightarrow_{\mu_n} Y\}$ a set of TFDs, with $XY \subseteq R$. Then for each module $\texttt{M} = (R, \mu, \phi)$, $\mathbf{Up}(\texttt{M}, cop(\mu, F))$ satisfies F iff M satisfies F.

Proof. The "if" ($\Leftarrow$) part is straightforward. Indeed, suppose $\mathbf{Up}(\texttt{M}, cop(\mu, F))$ does not satisfy F. Then there exist two tuples t_1 and t_2 such that $t_1 \in \phi'(j_1)$ and $t_2 \in \phi'(j_2)$ where ϕ' is the windowing function of $\mathbf{Up}(\texttt{M}, cop(\mu, F))$, j_1 and j_2 are such that $X \rightarrow Y$ is in $\pi_{cop(\mu,F)(j_1,j_2)}(F)$, and t_1 and t_2 agree on X but disagree on Y. Since each tick of $cop(\mu, F)$ is a composition of several ticks of μ, we have ticks i_1 and i_2 of μ such that $t_1 \in \phi(i_1)$ and $t_2 \in \phi(i_2)$ where ϕ is the windowing function of M, and $\mu(i_l) \subseteq cop(\mu, F)(j_l)$ for $l = 1, 2$. Thus, $X \rightarrow Y \in \pi_{\mu(i_1,i_2)}(F)$. This contradicts the fact that M satisfies F.

Now consider the "only-if" ($\Rightarrow$) part.

Since $cop(\mu, F)$ is obtained by recursively collapsing pairs of ticks provided that the two conditions are satisfied, it is sufficient to show that, at each step in the collapsing process, if a module M *raised* from the previous version of $cop(\mu, F)$ to the new $cop(\mu, F)$ satisfies F then M satisfies F. For simplicity we consider the first step, where λ is the type obtained by μ only collapsing two ticks.

Suppose by contradiction that $\texttt{M} = (R, \mu, \phi)$ does not satisfy F. Hence, there exists $X \longrightarrow_{\mu_k} Y$ in F that is not satisfied. By definition, there exist two tuples t_1 and t_2 and two non-empty ticks i_1 and i_2 of μ such that $t_1[X] = t_2[X]$, t_1 is in $\phi(i_1)$, t_2 is in $\phi(i_2)$, and there exists j such that $\mu(i_1, i_2) \subseteq \mu_k(j)$, but $t_1[Y] \neq t_2[Y]$. Since each tick of μ covered by a tick in at least one of the $\mu_1, \ldots, \mu_n$ is also covered by a tick of λ from its definition, there exist ticks k_1 and k_2 in λ such that $\mu(i_1) \subseteq \lambda(k_1)$ and $\mu(i_2) \subseteq \lambda(k_2)$.

Moreover, since for each k, we have $\phi'(k) = \bigcup_{i:\mu(i)\subseteq\lambda(k)} \phi(i)$, then $t_1 \in \phi'(k_1)$ and $t_2 \in \phi'(k_2)$. We now consider two cases: $k_1 = k_2$ and $k_1 \neq k_2$. In the first case both ticks i_1 and i_2 are covered by the same tick $k_1 = k_2$ of λ. In this case k_1 must be the tick obtained by collapsing and the two ticks of μ collapsed must be i_1 and i_2[7]. In this case, it is clear that $\lambda(k_1) \subseteq \mu_k(j)$. But then, since $\mathbf{Up}(\mathtt{M}, \lambda)$ satisfies F it satisfies also $X \longrightarrow_{\mu_k} Y$ and this is a contradiction since $t_1[Y] \neq t_2[Y]$. Consider the other case, i.e., $k_1 \neq k_2$. Then, if none of the ticks k_1 and k_2 of λ is the one obtained by collapsing, it means that $\lambda(k_1) = \mu(i_1)$ and $\lambda(k_2) = \mu(i_2)$. In this case obviously $\lambda(k_1, k_2) \subseteq \mu_k(j)$ and as for $k_1 = k_2$, we derive a contradiction. The only remaining case is when one of the ticks k_1, k_2 is obtained by collapsing. Suppose without loss of generality that $\mu(i_2, i_3) \subseteq \lambda(k_2)$ for some $i_3 \neq i_2$, $i_3 \neq i_1$. Then we know that $\mu(i_1) = \lambda(k_1)$. We also know that $\mu(i_1, i_2) \subseteq \mu_k(j)$. Now, if $\mu(i_3) \subseteq \mu_k(j)$ then $\lambda(k_1, k_2) \subseteq \mu_k(j)$ and as for previous cases, we derive a contradiction. Hence, $\mu(i_3) \not\subseteq \mu_k(j)$. In this case we know that the collapsing of ticks i_2 and i_3 by *cop*() implies the existence of a $\mu_r \neq \mu_k$ in $\{\mu_1, \ldots, \mu_n\}$ and of a tick r_1 such that $\mu(i_1, i_2, i_3) \subseteq \mu_r(r_1)$. The reason why even tick i_1 is covered by r_1 is that the condition used by *cop*() for collapsing imposes that no ticks of other types overlap with tick r_1. If tick r_1 would not cover i_1, tick j of μ_k would violate this condition. From $\mu(i_1, i_2, i_3) \subseteq \mu_r(r_1)$ we derive $\lambda(k_1, k_2) \subseteq \mu_r(r_1)$. However, since $\mathbf{Up}(\mathtt{M}, \lambda)$ satisfies F it satisfies also $X \longrightarrow_{\mu_r} Y$ and this is a contradiction since we have two tuples $t_1 \in \phi'(k_1)$ and $t_2 \in \phi'(k_2)$ such that $t_1[X] = t_2[X]$ and $t_1[Y] \neq t_2[Y]$.

We conclude that M must satisfy $X \longrightarrow_{\mu_k} Y$ and, since this reasoning applies to a generic TFD of F, it must satisfy F. This concludes the proof. □

Preservation of TFDs is in conflict with elimination of redundancy. Turning back to our example in the beginning of this section, let $\lambda = cop(\mathtt{day}, F)$, where $F = \{A \longrightarrow_{\mathtt{week}} B, A \longrightarrow_{\mathtt{month}} B\}$. Let d_1 be March 31, 1994 and d_2 April 1, 1994. Now suppose the tuple $t = (a, b)$ is in both $\phi(d_1)$ and $\phi(d_2)$. This is redundant. However, d_1 and d_2 belong to the same week, but they belong to two different months. Hence, by Figure 5, d_1 and d_2 are not combined together in calculating λ. Hence, these two tuples remain separate in the new module $\mathbf{Up}(\mathtt{M}, \lambda)$. If d_1 and d_2 were in one tick of λ, then the decomposition $\{(A, day), (AB, \lambda)\}$ of (AB, λ) would not preserve dependencies.

In the next section, we define the temporal analog of (non-temporal) 3NF that relaxes, in a restricted way, the conditions of TBCNF to allow certain types of data redundancy in order to preserve TFDs.

[7]Note that, at this step, only one tick was obtained by collapsing.

7 Temporal third normal form

Temporal third normal form (T3NF) is defined as follows:

Definition (Temporal 3NF)
A temporal module scheme (R, μ) with a set F of TFDs is in temporal third normal form (T3NF) if for each TFD $X \longrightarrow_{\nu} A$ that is logically implied by F, where (a) $XA \subseteq R$, (b) $A \notin X$, and (c) at least one tick of μ is covered by a tick of ν, one of the following conditions holds:

(i) A is part of a temporal candidate key of (R, μ), or

(ii) X is a temporal superkey of (R, μ), and there do not exist i_1 and i_2, with $i_1 \neq i_2$, such that $X \rightarrow A \in \pi_{\mu(i_1,i_2)}(F)$, unless there exists i_3, with $i_3 \neq i_1$, such that $X \rightarrow A \in \pi_{\mu(i_1,i_3)}(F)$ but $X \rightarrow A \notin \pi_{\mu(i_1,i_2,i_3)}(F)$.

Turning back to the temporal module scheme $(\mathtt{AB}, \mathtt{day})$ with $F = \{A \longrightarrow_{\mathtt{week}} B, A \longrightarrow_{\mathtt{month}} B\}$, we can easily see that $(A, \mathtt{day})$ and $(AB, cop(\mu, F))$ is a lossless T3NF decomposition of $(AB, \mathtt{day})$.

To see the difference between T3NF and TBCNF, we consider the temporal module scheme $(AB, \mathtt{day})$ with TFDs $A \longrightarrow_{\mathtt{day}} B$ and $B \longrightarrow_{\mathtt{month}} A$. This scheme is in T3NF and, thus, does not require any further decomposition. In contrast, since $B \longrightarrow_{\mathtt{month}} A$ holds, $(AB, \mathtt{day})$ is not in TBCNF. By the algorithm in Figure 4, it is decomposed into $(AB, \mathtt{month})$ and $(B, \mathtt{day})$. Although these schemes are in TBCNF, we can no longer enforce the TFD $A \longrightarrow_{\mathtt{day}} B$. In this case, the original scheme which is in T3NF may be preferred over the decomposed schemes in TBCNF.

7.1 Decomposing temporal module schemes into T3NF

In order to present the T3NF decomposition algorithm, we introduce the notions of subtype, partition of a type and union of subtypes in a partition.

A subtype of a type μ is intuitively a type having only a subset of the ticks of μ. For example, `Sunday` intended as a set of ticks corresponding to days that are Sundays, can be considered a subtype of `day`.

Definition We say that a type μ_1 is a *subtype* of a type μ if for each positive integer i, $\mu_1(i) = \mu(j)$ for some positive integer j.

A partition of a type μ is intuitively a set of disjoint subsets such that the set of all their ticks is the set of ticks of μ. Referring to the previous example, the set of the seven types {`Monday`,...,`Sunday`} is a partition of the type `day`.

Definition We say that a set of types $\{\mu_1, \ldots, \mu_n\}$ is a *partition* of type μ if:

- each μ_i with $1 \leq i \leq n$ is a subtype of μ
- for each positive integer l, $\mu(l) = \mu_k(j)$ for some k with $1 \leq i \leq n$ and positive integer j. Moreover $\mu_k(j) \neq \mu_i(r)$ for each $i \neq k$ and positive integer r.

We can also take the union of two subtypes of the same type generating a new subtype. For example, `Saturday` $\cup$ `Sunday` is the type whose ticks correspond to days that are only Saturdays and Sundays. This can be useful if we want to store information concerning only these two days, but still being able to relate this information with other information taken for different days.

Definition Given types μ_1 and μ_2 that are both subtypes of μ we define the *union* of them as $\mu_1 \cup \mu_2$ such that for each tick of $\mu_1 \cup \mu_2$ there exists a tick of μ_1 or of μ_2 that is equal to it, and conversely, for each tick of μ_1 and of μ_2 there exists a tick of $\mu_1 \cup \mu_2$ that is equal to it. That is, for all i, $(\mu_1 \cup \mu_2)(i) = \mu_1(j_1)$ or $(\mu_1 \cup \mu_2)(i) = \mu_2(j_2)$ for some j_1 and j_2, and for all j_1 and j_2, $\mu_1(j_1) = (\mu_1 \cup \mu_2)(i_1)$ and $\mu_2(j_2) = (\mu_1 \cup \mu_2)(i_2)$ for some i_1 and i_2.

Note that the union of two subtypes of μ is still a subtype of μ and the set obtained by taking a partition of μ and replacing two subtypes with their union is still a partition of μ.

Similar to the traditional 3NF decomposition [9], we use temporal minimal cover for the T3NF decomposition algorithm.[8]

Definition (Temporal Minimal Cover)
Let F be a set of TFDs and $NTC(F)$ the traditional non-temporal minimal cover of $\pi_{\emptyset}(F)$. We say that G is a *(temporal) minimal cover* of F if $G = \{X \longrightarrow_{\mu} A \mid (A, \mu) \in \overline{X}^{+} \text{ and } X \rightarrow A \in NTC(F)\}$.

As an example, consider the temporal types and TFDs given in Example 3. Clearly, $NTC(F) = \pi_{\emptyset}(F) = \{A \rightarrow B, B \rightarrow A\}$. By definition, the following set is a temporal minimal cover of F: $\{A \longrightarrow_{\texttt{wr}} B, B \longrightarrow_{\texttt{month}} A\}$, which is F itself as expected.

We are now ready for the T3NF decomposition algorithm.

[8] For those not familiar with the non-temporal minimal cover, a *minimal cover* of a set F of FDs is a set F_c of FDs that is equivalent to F and satisfies the following three conditions [9]: (i) Every right side of a dependency in F_c is a single attribute; (ii) For no $X \rightarrow A$ in F_c is the set $F_c - \{X \rightarrow A\}$ equivalent to F; and (iii) for no $X \rightarrow A$ in F_c and proper subset Z of X is $(F - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ equivalent to F.

Algorithm for T3NF Decomposition

INPUT: A temporal module scheme (R, μ) and a set F of functional dependencies.

OUTPUT: A lossless decomposition $\rho = (R_1, \lambda_1), \ldots, (R_n, \lambda_n)$ such that each (R_i, λ_i) is in T3NF.

METHOD:

1. Let $\mathbf{MinCov}(F)$ be a temporal minimal cover of F. From the set of all types $\{\nu_1, \ldots, \nu_l\}$ appearing in the TFDs of $\mathbf{MinCov}(F)$, compute a partition $P = \{\mu_1, \ldots, \mu_m\}$ of μ as follows.

   Starting with $P^{(0)} = \{\mu\}$, for each ν_i in the TFDs starting from $i = 1$ to $i = l$, get $P^{(i)}$ substituting each μ_k in $P^{(i-1)}$ with μ_{k1}, μ_{k2}, where μ_{k1} is obtained by μ_k dropping all nonempty ticks contained in ν_i and μ_{k2} is the complementary type, i.e., the type having only those ticks. $P = P^{(l)}$ will be the desired partition. For each nonempty tick j of μ, there is one and only one type μ_k in $P^{(l)}$ with a tick s covering it. Moreover, $\mu(j) = \mu_k(s)$.

   For each μ_k in P define the set[a]

   $F_{\mu_k} = \{X \longrightarrow_{\nu} A \mid X \longrightarrow_{\nu} A \in \mathbf{MinCov}(F)$ and $\exists j, i \;\; \mu_k(j) \neq \emptyset \;\wedge\; \mu_k(j) \subseteq \nu(i)\}$.

2. Let $\rho = \emptyset$. For each $X \rightarrow A$ in $\pi_{\emptyset}(\mathbf{MinCov}(F))$, given

   - $\{\mu_r, \ldots, \mu_s\} = \{\mu_k \in P \mid \exists \nu_i$ s.t. $X \longrightarrow_{\nu_i} A \in F_{\mu_k}\}$ and
   - $F_{X \rightarrow A} = \{X \longrightarrow_{\nu_i} A \mid \exists \mu_k \in P$ s.t. $X \longrightarrow_{\nu_i} A \in F_{\mu_k}\}$,

   add $(XA, cop(\mu_r \cup \cdots \cup \mu_s, F_{X \rightarrow A}))$ to ρ.

3. For each μ_k in P add (Z, μ_k) to ρ where Z is a temporal candidate key of (R, μ_k), if there is no (V, μ_r) in ρ, with $Z \subseteq V$ and μ_k a subtype of μ_r.

4. If (X, ν) and (Y, ν) with $X \subseteq Y$ are both in ρ, drop (X, ν) from ρ. Furthermore, if (X, ν_1) and (X, ν_2) are both in ρ and ν_1 and ν_2 are both subtypes of a temporal type, drop (X, ν_1) and (X, ν_2) but add $(X, \nu_1 \cup \nu_2)$. This last step may be repeated until no change can be made.

[a] These sets are easily obtained by simply associating to μ_{k2} the TFDs with type ν_i at each step of calculating P.

Figure 6: Algorithm for T3NF decomposition.

Theorem 6 The algorithm in Figure 6 always terminates and gives a lossless T3NF decomposition of the input temporal module scheme. Furthermore, the decomposition preserves dependencies.

A proof is supplied in Appendix A.3.

We illustrate the T3NF decomposition algorithm by continuing the example given before Theorem 6. Assume $(AB, \mathtt{day})$ is our temporal module scheme. The minimal cover of F is F itself. Therefore, $\pi_{\emptyset}(\mathbf{MinCov}(F)) = \{A \to B, B \to A\}$. For step 1, we use the two temporal types: $\mathtt{month}$ and $\mathtt{w_r}$ to partition temporal type $\mathtt{day}$. Here, $\mathtt{day}$ is partitioned into two types: $\mathtt{d_p}$ and $\mathtt{d_r}$, i.e., the part days (the days before July 4, 1994) and the recent days (the days on and after July 4, 1994). Thus, $F_{\mathtt{d_p}} = \{B \longrightarrow_{\mathtt{month}} A\}$ and $F_{\mathtt{d_r}} = F$. Now for $A \to B$ and $B \to A$ in $\pi_{\emptyset}(\mathbf{MinCov}(F))$, we create the module schemes $(AB, cop(\mathtt{d_r}, F_{A\to B})$ and $(AB, cop(\mathtt{d_p} \cup \mathtt{d_r}, F_{B\to A}))$, respectively. Note that $cop(\mathtt{d_r}, F_{A\to B}) = \mathtt{w_r}$ and $cop(\mathtt{d_p} \cup \mathtt{d_r}, F_{B\to A}) = \mathtt{month}$. Hence, we have created the module schemes $(AB, \mathtt{w_r})$ and $(AB, \mathtt{month})$. As the final step, we create $(B, \mathtt{d_p})$ and $(B, \mathtt{d_r})$, which is combined into one temporal scheme $(B, \mathtt{day})$. In summary, the T3NF decomposition of $(AB, \mathtt{day})$ is $(B, \mathtt{day})$, $(AB, \mathtt{w_r})$ and $(AB, \mathtt{month})$. All three new schemes are in T3NF and the decomposition is lossless and preserves dependencies.

8 Conclusion

This paper introduced temporal functional dependencies, a type of natural temporal constraint. To reduce data redundancy arising from these dependencies, temporal normal forms and their decomposition algorithms were given. These normal forms and algorithms are proper extensions of the traditional normal forms and algorithms in that if all data are in one tick of a temporal type, then the temporal normal forms and their algorithms reduce to their non-temporal counterparts.

To build effective procedures for the algorithms in the paper, certain operations on temporal types must be effective. In the algorithm for $\overline{X}^+$, we needed to calculate the *glb* of a finite number of temporal types; also, we needed to know if $\mu \prec \nu$ for given temporal types μ and ν. In the TBCNF decomposition algorithm, given two temporal types μ and ν, we needed temporal types ν_1, ν_2 and ν_3 that are basically obtained from a partition of temporal types and from combining certain ticks of a give temporal type. Finally, in the T3NF algorithm, we needed to compute the *cop* function. Because our set of temporal types is uncountably infinite, it does not yield effective procedures for these operations. However, for realistic systems, many (infinite) temporal types can be finitely described in a way that these operations are effective. For example, for periodic temporal types[9] above operations do have effective procedures.

[9] A temporal type is periodic if after certain ticks, each tick is only a shift (of a fixed length) of some previous tick.

Note that the every-day temporal types like `week`, `month`, etc. are all periodic.

A basic structure used in the paper is the temporal modules. It is important to emphasize that the temporal modules are rather general. Results of the paper are readily applicable to all temporally ungrouped models [1]. It will be interesting, as a future research problem, to apply the concepts and methods to temporally grouped relational models [1].

Our work is set in the framework of a particular temporal type system. Since a temporal type is a monotonic mapping from positive integers to the power set of the real numbers, in this type system, the time is always positive, and every tick of a temporal type consists of an arbitrary set of real numbers. These choices, however, are not entirely inherent to the results presented in the paper; they are motivated by our desire for a simpler and intuitively appealing presentation of the results. The results of this paper hold for a more general definition of temporal type using all the integers to represent time ticks. In fact, the only requirement on the temporal types we used in the paper is the following: A temporal type is a set of pairwise non-intersecting sets. The results of the paper should hold with any type system with this property.

Temporal types generated by the TBCNF and T3NF decomposition algorithms may be quite complex and not intuitive to users. This problem can be solved however by implementing a conceptual database level that allow the users to view the data, update and pose queries assuming that the data is in terms of basic temporal types they intuitively understand. At the lower level, transparent to the users, the underlying implementation may use complex temporal types to facilitate the removal of data redundancy. This idea is similar to views of traditional relational databases. The exact understanding and implementation of this idea requires further investigation.

Finally, we note that data redundancy exists in spatial data, for example, in geographic information systems due to constraints similar to TFDs. To understand the analogy, let a *spatial type* be defined as a static collection of spatial objects, like `states`, `counties`, `cities`, etc. Consider a spatial relation (`Company, Branch-Coordinates, Supplies-Contractor`) with a *spatial* FD `Company` $\longrightarrow_{\texttt{state}}$ `Supplies-Contractor`, which intuitively states that within a state boundary, all branches of a company have to use the same supplies contractor. Since the same supplies contractor is repeated for each branch in the same state, we have data redundancy. We may apply a technique that is very similar to what we have developed for the temporal case, and obtain the decomposition (`Company, Branch-Coordinates`) and (`State, Supplies-Contractor`).

Formally speaking, a temporal type μ is *periodic* if there exist M and m such that for each $k \geq M$, $\mu(k) = \mu(k - m)$.

References

[1] J. Clifford, A. Croker and A. Tuzhilin. On the completeness of query languages for grouped and ungrouped historical data models. *ACM Transactions on Database Systems*, 19(1):64–116. March 1994.

[2] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, Great Britain, 1990.

[3] C. S. Jensen, R. T. Snodgrass, and Michael D. Soo. Extending normal forms to temporal relations. Technical Report TR 92–17, University of Arizona, July 1992.

[4] F. Kabanza, J-M. Stevenne and P. Wolper. Handling infinite temporal data. In *Proceedings of 9th ACM Symposium on Principles of Database Systems.* Nashville, Tennessee. April, 1990.

[5] N. Lorentzos and V. Kollias. The handling of depth and time intervals in soil-information systems. *Computers and Geosciences*, 15(3):395–401, 1989.

[6] S. B. Navathe and R. Ahmed. A temporal relational model and a query language. *Information Sciences*, 49:147–175, 1989.

[7] M. Niezette and J. Stevenne. An Efficient Symbolic Representation of Periodic Time. *First International Conference on Information and Knowledge Management.* Baltimore, MD. November, 1992.

[8] A. Segev and A. Shoshani. The representation of a temporal data model in the relational environment. In *Proceeding of the 4th International Conference on Statistical and Scientific Database Management*, 1988.

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

[10] V. Vianu. Dynamic functional dependencies and database aging. *JACM*, 34(1):28–59, 1987.

[11] X.S. Wang. An algebraic query language on federated temporal databases. Technical Report ISSE-TR-94-107. Department of Information and Software Systems Engineering, George Mason University. June 1994.

[12] X.S. Wang, S. Jajodia, and V.S. Subrahmanian. Temporal modules: An approach toward federated temporal databases. In *Proceedings of 1993 ACM SIGMOD International Conference on the Management of Data*, Washington, D.C., 1993.

[13] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with granularity of time in temporal databases. In *Proc. 3rd Nordic Conf. on Advanced Information Systems Engineering*, Trondheim, Norway, may 1991.

Appendix

A.1 Proof of Theorem 3

Proof.

Termination. Let $\{\mu_1, \ldots, \mu_n\}$ be the set of types appearing in F and $U_T = \{(A, \nu) \mid A \in U \text{ and } \nu = glb(\mu_s, \ldots, \mu_t) \text{ where } \{\mu_s, \ldots, \mu_t\} \subseteq \{\mu_1, \ldots, \mu_n, \mu_{\mathrm{Top}}\}\}$. U_T is a finite set, since U is finite and the set of types appearing in F is also finite. Since $X^{(0)} \subseteq \ldots \subseteq X^{(i)} \subseteq \ldots \subseteq U_T$ we must eventually reach i such that $X^{(i)} = X^{(i+1)}$.

Correctness. Let Z be the set returned by the algorithm. We need to show that $Z = \overline{X}^+$. Suppose i is the integer in the algorithm such that $\overline{X}^{(i)} = \overline{X}^{(i+1)}$. We first show the following:

A. $(B, \mu) \in \overline{X}^{(i)}$ implies $X \longrightarrow_{\mu} B \in \overline{F}^+$

We prove by induction on the number j of iterations of step 2 in the algorithm that if $(B, \mu) \in X^{(j)}$, for some $j \geq 0$ then $X \longrightarrow_{\mu} B \in \overline{F}^+$.

Basis: For $j = 0$ the set contains only pairs $(A_i, \mu_{\mathrm{Top}})$ such that $A_i \in X$. If $A_i \in X$ then, by restricted reflexivity, $F \vdash_{\mathrm{f}} X \longrightarrow_{\mu_{\mathrm{Top}}} A_i$ and, hence $X \longrightarrow_{\mu_{\mathrm{Top}}} A_i \in \overline{F}^+$.

Induction: Let $j > 0$ and suppose this is true for j. We want to prove that if $(B, \mu) \in X^{(j+1)}$ then $X \longrightarrow_{\mu} B \in \overline{F}^+$. The algorithm computes $X^{(j+1)}$ in step 2. If (B, μ) is also in $X^{(j)}$ then, by the induction hypothesis, $X \longrightarrow_{\mu} B \in \overline{F}^+$. Otherwise, $(B, \mu) \in X^{(j+1)}$, but $(B, \mu) \notin X^{(j)}$. From step 2, $(B, \mu) \in Y$. Since $(B, \mu) \in Y$, it must be in one of $Y_1, \ldots, Y_r$ and therefore, there exists a TFD $A_1 \ldots A_k \longrightarrow_{\mu'} B_1 \ldots B_m$ in F such that $\{(A_1, \mu_1), \ldots, (A_k, \mu_k)\}$ is a subset of $X^{(j)}$ and B is one of the $B_1 \ldots B_m$. Moreover $\mu = glb(\mu_1, \ldots, \mu_k, \mu')$. By induction hypothesis, $F \vdash X \longrightarrow_{\mu_i} A_i$ for each $1 \leq i \leq k$. By union rule, $F \vdash X \longrightarrow_{glb(\mu_1, \ldots, \mu_k)} A_1 \ldots A_k$. By extended transitivity, $F \vdash X \longrightarrow_{glb(\mu_1, \ldots, \mu_k, \mu')} B_1 \ldots B_m$. By restricted reflexivity, $B_1 \ldots B_m \longrightarrow_{\mu_{\mathrm{Top}}} B$ and, finally, by extended transitivity, $X \longrightarrow_{glb(\mu_1, \ldots, \mu_k, \mu')} B$. Since restricted reflexivity and extended transitivity are Finite

Inference Axioms and the *union rule* is derived by augmentation and extended transitivity that are both Finite Inference Axioms, we have $F \vdash_{\mathrm{f}} X \longrightarrow_{glb(\mu_1,\ldots,\mu_k,\mu')} B$. Since $\mu = glb(\mu_1, \ldots, \mu_k, \mu')$, by definition of $\overline{F}^+$, $X \longrightarrow_\mu B \in \overline{F}^+$ concluding the induction proof. Applying the result proved by induction with $X^{(j)} = X^{(i)}$ we obtain that $(B, \mu) \in X^{(i)}$, where $X^{(i+1)} = X^{(i)}$, implies $X \longrightarrow_\mu B \in \overline{F}^+$.

We now prove the following claim:

B. If $F \vdash X \longrightarrow_\mu B$ and $\mu = glb(F')$ where F' is a minimal support for $X \rightarrow B$, then (B, μ) is in $X^{(i)}$.

Since F' is a minimal support for $X \rightarrow B$, by the standard completeness theorem, we know that $\pi_\emptyset(F') \vdash X \rightarrow B$. By the minimality of F' we know that all of the dependencies in $\pi_\emptyset(F')$ are used in the derivation. Since each FD is used, we know that if $C_1 \ldots C_r \rightarrow W$ is a FD in $\pi_\emptyset(F')$ then either $C_i \subseteq X$ or there exists a FD $V \rightarrow C_i Y$ in $\pi_\emptyset(F')$. Indeed, if this is not the case $C_1 \ldots C_r \rightarrow W$ cannot be used in the derivation of $X \rightarrow B$. Let , (F') be a sequence of the FDs in $\pi_\emptyset(F')$ such that:
$C_1 \ldots C_r \rightarrow W \in$, (F') and $C_i \not\subseteq X$ with $1 \leq i \leq r$
implies that
$V \rightarrow C_i Y \in$, (F') and $V \rightarrow C_i Y$ precedes $C_1 \ldots C_r \rightarrow W$ in , (F').
We prove by induction on the length n of , (F') that if $F \vdash X \longrightarrow_\mu B$, F' is a minimal support for $X \rightarrow B$, and $\mu = glb(F')$ then (B, μ) is in $X^{(n)}$.
If $n = 0$ it means that $F' = \emptyset$. This implies $\mu = \mu_{\mathrm{Top}}$, and $B \subseteq X$. In this case we know that (B, μ_{Top}) is in $X^{(0)}$. Assume this is true for $0 < n \leq q$. We prove it for $q + 1$. The last element of , (F') must be a functional dependency $C_1 \ldots C_r \rightarrow W$ such that $B \in W$. Consider the subsequence , $^-$ obtained by dropping $C_1 \ldots C_r \rightarrow W$ from , (F'). By the ordering on the sequence we know that for each C_i with $1 \leq i \leq r$ either $C_i \subseteq X$ or there exists $V \rightarrow C_i Y$ in , $^-$. If $C_i \subseteq X$ we know that $(C_i, \mu_{\mathrm{Top}}) \in X^{(0)}$ and hence $(C_i, \mu_{\mathrm{Top}}) \in X^{(q)}$. If $V \rightarrow C_i Y \in$, $^-$, there exists a minimal support $F_i \subseteq F'$ for $X \rightarrow C_i$ such that all the FDs in $\pi_\emptyset(F_i)$ are in , $^-$. Hence, by induction hypothesis, we have that $(C_i, \mu_i) \in X^{(q)}$ with $\mu_i = glb(F_i)$. Since $\pi_\emptyset(F')$ is derived from F', there exists a TFD $C_1 \ldots C_r \longrightarrow_{\mu'} W$ in F' for some temporal type μ'. Since there is a (C_i, μ_i) in $X^{(q)}$ for each $1 \leq i \leq r$, at step $q + 1$ the algorithm must consider the TFD $C_1 \ldots C_r \longrightarrow_{\mu'} W$ and the pair (B, μ) with $\mu = glb(\mu_1, \ldots, \mu_r, \mu')$ will appear in $X^{(q+1)}$. Note that $\mu_i = glb(F_i)$ for $1 \leq i \leq r$ and we know that $F_i \subseteq F'$. Hence, $F_1 \cup \ldots F_r \cup \{C_1 \ldots C_r \longrightarrow_{\mu'} W\} \subseteq F'$. Then $glb(F') \preceq glb(\mu_1, \ldots, \mu_r, \mu') = \mu$. However, since F' is a minimal support we know that the algorithm must have used all the TFDs in F'. Hence it must be $\mu \preceq glb(F')$ and using $glb(F') \preceq \mu$ we derive $\mu = glb(F')$. This concludes the induction proof.

As the third step of the proof, we show

C. If $X \longrightarrow_\nu B$ is in $\overline{F}^+$, then $\nu \preceq glb(F_i)$, for some minimal support F_i for $X \rightarrow B$.

Suppose by contradiction that $\nu \not\preceq glb(F_i)$ for all minimal support F_i for $X \rightarrow B$. Let $F_1, \ldots, F_m$ be all the minimal supports for $X \rightarrow B$. Then for each $1 \leq i \leq m$, there exist j and $V_i \longrightarrow_{\mu_i} W_i$ in F_i such that $\emptyset \neq \nu(j) \not\subseteq \mu_i(k)$ for all k. Thus, for all $1 \leq i \leq m$, there exists j such that $\pi_\emptyset(F_i) \not\subseteq \pi_{\nu(j)}(F)$, where $\pi_{\nu(j)}(F) = \{V \longrightarrow_\lambda W \in F | \nu(j) \subseteq \lambda(l) \text{ for some } l\}$. Since $X \longrightarrow_\nu B$ is in $\overline{F}^+$, by the Theorem 1, we know $F \models X \longrightarrow_\nu B$. Hence, by Lemma 2, for all j such that $\nu(j) \neq \emptyset$, we have $\pi_{\nu(j)}(F) \vdash X \rightarrow B$. Hence, there exists a minimal support F' for $X \rightarrow B$ such that $\pi_\emptyset(F') \subseteq \pi_{\nu(j)}(F)$. However, we know that each $\pi_\emptyset(F_i) \not\subseteq \pi_{\nu(j)}(F)$ for all the minimal support F_i for $X \rightarrow B$. This is a contradiction. Therefore, $\nu \preceq glb(F_i)$ for some minimal support F_i for $X \rightarrow B$.

As the fourth step of the proof, we show the following:

D. $Z \subseteq \overline{X}^+$.

Assume by contradiction that there exists $(B, \mu) \in Z$ but $(B, \mu) \notin \overline{X}^+$. Since (B, μ) is in Z and hence in $\overline{X}^{(i)}$, we have $X \longrightarrow_\mu B \in \overline{F}^+$ by **A**. By **C**, we know that exists a minimum support F_i for $X \rightarrow B$ such that $\mu \preceq glb(F_i)$. We also know that $\mu \not\prec glb(F_i)$, since otherwise, (B, μ) is removed because $(B, glb(F_i))$ is in $\overline{F}^+$ for all minimal support F_i for $X \rightarrow B$ by **B**. Since $\mu \preceq glb(F_i)$ and $\mu \not\prec glb(F_i)$, we know $\mu = glb(F_i)$. Now since $(B, \mu) \notin \overline{X}^+$ but $X \longrightarrow_\mu B \in \overline{F}^+$, by the definition of $\overline{X}^+$, we know there exists ν such that $X \longrightarrow_\nu B$ is in $\overline{F}^+$ and hence, by **C**, there exists a minimal support F_j for $X \rightarrow B$ such that $\nu \preceq glb(F_j)$. Hence, $\mu \prec \nu \preceq glb(F_j)$. This is a contradiction since we know that $\mu \not\prec glb(F')$ for all minimal support F' for $X \rightarrow B$. Therefore, **D** holds.

As the last step, we establish:

E. $\overline{X}^+ \subseteq Z$.

We first show $\overline{X}^+ \subseteq \overline{X}^{(i)}$. Assume (B, μ) is in $\overline{X}^+$. By definition, $X \longrightarrow_\mu B$ is in $\overline{F}^+$ and there exists no λ with $\mu \prec \lambda$ such that $X \longrightarrow_\lambda B$ is in $\overline{F}^+$. Hence, $\mu \preceq glb(F_i)$ for some minimal support F_i for $X \rightarrow B$ by **C**. We know $\mu \not\prec glb(F_i)$. Indeed, since $(B, glb(F_i))$ is in $\overline{X}^{(i)}$ and by **A**, we know $X \longrightarrow_{glb(F_i)} B$ is in $\overline{F}^+$. Hence, (B, μ) cannot be in $\overline{X}^+$ by definition if $\mu \prec glb(F_i)$. Therefore, $\mu = glb(F_i)$. Since we know $(B, \mu) = (B, glb(F_i))$ is in $\overline{X}^{(i)}$ by **B**, we have $\overline{X}^+ \subseteq \overline{X}^{(i)}$ because (B, μ) is a generic element of $\overline{X}^+$. Now we show **E**. Assume by contradiction that there exists (B, μ) in $\overline{X}^+$ but not in Z. Since $(B, \mu) \in \overline{X}^+$, we have (B, μ) in $\overline{X}^{(i)}$ since $\overline{X}^+ \subseteq \overline{X}^{(i)}$. By the fact that (B, μ) is not in Z and (B, μ) is in $\overline{X}^{(i)}$, we know there exists ν such that (B, ν) in Z, with $\mu \prec \nu$, by the definition of Z, and hence in $\overline{X}^{(i)}$. By **D**, we know (B, ν) is in $\overline{X}^+$. This is a contradiction since (B, ν) is in $\overline{X}^+$ and (B, μ) is also in $\overline{X}^+$ with $\mu \prec \nu$. Hence, we have shown $\overline{X}^+ \subseteq Z$.

By combining **D** and **E**, we know that the theorem holds. □

A.2 Proof of Theorem 5

Before showing the correctness of the above algorithm, we first present the following preliminary result.

Lemma 4 Let (R, μ) be a temporal module and F a set of TFDs.
Let $\rho_1 = \{(R_1, \mu_1), \ldots, (R_n, \mu_n)\}$ be a tickwise-lossless decomposition of (R, μ) wrt F, $\rho_2 = \{(R_{11}, \mu_{11}), \ldots, (R_{1m}, \mu_{1m})\}$ a tickwise-lossless decomposition of (R_1, μ_1) wrt F, and $\rho = \{(R_{11}, \mu_{11}), \ldots, (R_{1m}, \mu_{1m}), (R_2, \mu_2), \ldots, (R_n, \mu_n)\}$. If for all ticks k of μ the following hold:

(a) for all k_1, $\mu(k) \subseteq \mu_1(k_1)$ implies $\mathbf{MaxSub}(\mu_1(k_1), \rho_2) = \rho_2 \cap \mathbf{MaxSub}(\mu(k), \rho)$, and

(b) $\rho_2 \cap \mathbf{MaxSub}(\mu(k), \rho) \neq \emptyset$ iff $(R_1, \mu_1) \in \mathbf{MaxSub}(\mu(k), \rho_1)$,

then ρ is a tickwise-lossless decomposition of (R, μ) wrt F.

Proof. Let k be a non-empty tick of μ and $\mathtt{M} = (R, \mu, \phi)$ be a temporal module that satisfies F. Two cases arise:

i. $\mathbf{MaxSub}(\mu(k), \rho) \cap \rho_2 = \emptyset$, and

ii. $\mathbf{MaxSub}(\mu(k), \rho) \cap \rho_2 \neq \emptyset$.

For each case, we need to show $\phi(k)$ is a join of the modules from $\mathbf{MaxSub}(\mu(k), \rho)$. Consider case i. In this case, $\mathbf{MaxSub}(\mu(k), \rho)$ is a subset of $\{(R_2, \mu_2), \ldots, (R_n, \mu_n)\}$. Also, by the hypothesis (b) of the lemma, (R_1, μ_1) is not in $\mathbf{MaxSub}(\mu(k), \rho_1)$. Then the tickwise-lossless property of ρ_1 ensures that $\phi(k)$ can be recovered by the join of the windowing functions in the decomposition ρ_1 and in this case, it is the same join for the decomposition ρ. Thus, $\phi(k)$ is recovered from the the join of the modules on the schemes in $\mathbf{MaxSub}(\mu(k), \rho)$. Consider case ii. Since some of the schemes in ρ_2 belong to $\mathbf{MaxSub}(\mu(k), \rho)$, (R_1, μ_1) must be included in $\mathbf{MaxSub}(\mu(k), \rho_1)$ by the hypothesis (b) of the lemma. Since (R_1, μ_1) is in $\mathbf{MaxSub}(\mu(k), \rho_1)$, there exists a tick k_1 of μ_1 covering tick k of μ, i.e., $\mu(k) \subseteq \mu_1(k_1)$. By hypothesis (a), the set of schemes of ρ_2 present in $\mathbf{MaxSub}(\mu(k), \rho)$ is exactly $\mathbf{MaxSub}(\mu_1(k_1), \rho_2)$. Thus we can assume that $\mathbf{MaxSub}(\mu_1(k_1), \rho_2) = \{(S_1, \nu_1), \ldots, (S_m, \nu_m)\}$,

$$\mathbf{MaxSub}(\mu(k), \rho) = \{(S_1, \nu_1), \ldots, (S_m, \nu_m), (S'_1, \nu'_1), \ldots, (S'_n, \nu'_n)\},$$

and $\mathbf{MaxSub}(\mu(k), \rho_1) = \{(R_1, \mu_1), (S'_1, \nu'_1), \ldots, (S'_n, \nu'_n)\}$. Let $\mathtt{M}_{R_1} = (R_1, \mu_1, \phi_{R_1}) = \mathbf{Up}(\pi^T_{R_1}(\mathtt{M}), \mu_1)$, $(S_j, \nu_j, \phi_{S_j}) = \mathbf{Up}(\pi^T_{S_j}(\mathtt{M}_{R_1}), \nu_j)$ for each $1 \leq j \leq m$, and $(S'_j, \nu'_j, \phi_{S'_j}) = \mathbf{Up}(\pi^T_{S'_j}(\mathtt{M}), \nu'_j)$ for each

$1 \leq j \leq n$. Also, let k_{S_j} be the integer such that $\mu_1(k_1) \subseteq \nu_j(k_{S_j})$ for each $1 \leq j \leq m$ and $k_{S'_j}$ the integer such that $\mu(k) \subseteq \nu'_j(k_{S'_j})$ for each $1 \leq j \leq n$. Since ρ_1 is a tickwise lossless decomposition of (R, μ) wrt F, we have $\phi(k) = \phi_1(k_1) \bowtie \phi_{S'_1}(k_{S'_1}) \bowtie \cdots \bowtie \phi_{S'_n}(k_{S'_n})$. Furthermore, since ρ_2 is a tickwise lossless decomposition of (R_1, μ_1) wrt F, we have $\phi_1(k_1) = \phi_{S_1}(k_{S_1}) \bowtie \cdots \bowtie \phi_{S_m}(k_{S_m})$. Thus, we have $\phi(k) = \phi_{S_1}(k_{S_1}) \bowtie \cdots \bowtie \phi_{S_m}(k_{S_m}) \bowtie \phi_{S'_1}(k_{S'_1}) \bowtie \cdots \bowtie \phi_{S'_n}(k_{S'_n})$. The only thing left to be shown is that $\mu(k) \subseteq \nu_j(k_{S_j})$ for each $1 \leq j \leq m$. However, this is clear since $\mu_1(k_1) \subseteq \nu_j(k_{S_j})$ and $\mu(k) \subseteq \mu_1(k_1)$. $\square$

We are now ready to show the correctness of the TBCNF decomposition algorithm.

Proof. Let (R, μ) be the input scheme and F the input set of TFDs. We now show that the algorithm always terminates. To do this, we associate to each scheme (R_i, μ_i) a tuple of three non-negative integers (a, c, n), called its *index*, where

- a denotes the number of attributes in the scheme,
- c is the number of TFDs $V \longrightarrow_\nu W$ in $\overline{\pi}_{R_i}(F)$ such that there exist tick k_1 and k_2 of μ_i that is covered by some tick l of ν, i.e., $\mu_i(k_1, k_2) \subseteq \nu(l)$, and
- n is the number of TFDs $V \longrightarrow_\nu W$ in $\overline{\pi}_{R_i}(F)$ such that there exist ticks k_1 and k_2 such that $\mu_i(k_1) \subseteq \nu(j_1)$ for some j_1, and $\mu_i(k_2) \not\subseteq \nu(j_2)$ for all j_2. That is, n is the number of TFDs whose temporal type cover a proper subset of ticks of μ_i. We call such TFDs *partial TFDs wrt* μ_i.

It is easily seen that if $a = 1$, then the scheme is in TBCNF. We say that $(a_1, c_1, n_1) \geq (a_2, c_2, n_2)$ if $a_1 \geq a_2$, $c_1 \geq c_2$ and $n_1 \geq n_2$. If $(a_1, c_1, n_1) \geq (a_2, c_2, n_2)$ and at least one of these corresponding numbers are not equal, i.e., either $a_1 > a_2$, $c_1 > c_2$ or $n_1 > n_2$, then we say that (a_1, c_1, n_1) is larger than (a_2, c_2, n_2). At each step, the algorithm selects a scheme (R_i, μ_i) and a TFD $X \longrightarrow_\nu A$ in $\overline{\pi}_{R_i}(F)$ that violates the TBCNF condition for that scheme. The scheme (R_i, μ_i) is then replaced by 3 schemes. We claim that the index for (R_i, μ_i) is (strictly) larger than that for each of the new schemes. Clearly, if this claim holds, and by the fact that the index for each scheme is $\geq (1, 0, 0)$, we can then conclude that the algorithm always terminates in a finite number of steps.

Let us turn to establish our claim. Consider each of the three schemes that replaces (R_i, μ_i). Assume that the index for (R_i, μ_i) is (a, c, n). Suppose the index for one of the three new schemes is (a', c', n'). It is easily seen that $a \geq a'$, $c \geq c'$ and $n \geq n'$. We now show that (a, c, n) is strictly larger than (a', c', n') by considering each of the three new schemes:

1. Suppose (a', c', n') is the index for (R_i, ν_1). Clearly, $a' = a$ and $c' \leq c$. It is also easily seen that $n' < n$. Indeed, by the algorithm, the scheme (R_i, ν_1) exists only if there is at least one non-empty tick of ν_1, and also each tick of ν_1 is not covered by any tick of ν. [Note that $X \longrightarrow_\nu A$ is the TFD that violates the TBCNF condition in (R_i, μ_i) and is used to decompose (R_i, μ_i).] Since $X \longrightarrow_\nu A$ violates the TBCNF condition in (R_i, μ_i), it implies that there exists at least one tick of μ_i that is covered by a tick of ν. Thus, $X \longrightarrow_\nu A$ is a partial TFD wrt μ_i. However, this TFD is not partial wrt ν_1. Also, it is clear, that a non-partial TFD wrt μ_i is still a non-partial TFD wrt ν_1. Hence, the number of partial TFDs wrt ν_1 is strictly less than the number of partial TFDs wrt μ_i, i.e., $n' < n$.

2. Suppose (a', c', n') is the index for $(R_i - A, \nu_2)$. Clearly, $a' = a - 1$, and hence $a > a'$. It is also clear that $c > c'$ and $n > n'$. Hence, $(a, c, n) > (a', c', n')$.

3. Suppose (a', c', n') is the index for (XA, ν_3). Two cases arise: $a > a'$ and $a = a'$. In the first case, we have $(a, c, n) > (a', c', n')$ as desired. Suppose now that $a = a'$. We consider two subcases: $c > c'$ or $c = c'$. Again, if the first subcase holds, then we are done. So suppose $c = c'$. It is easily seen that no two distinct ticks of μ_i are covered by a single tick of ν. Indeed, assuming otherwise, i.e., by the construction of ν_3, there exists a tick of ν_3 that covers two distinct ticks k_1 and k_2 of ν_2. Hence, there exists a tick of ν that covers both ticks k_1 and k_2 of μ_i. However, no tick of ν covers two distinct ticks of ν_3 by the construction of ν_3. Thus, $c > c'$, a contradiction. Now we have $a = a'$ and $\nu_2 = \nu_3$. Then $XA = R_i$. However, we know that $X \longrightarrow_\nu A$ violates the TBCNF condition for (R_i, μ_i). This implies only two possibilities: (i) X is not a temporal superkey of (R_i, μ_i), or (ii) there are two distinct ticks of μ_i that are covered by a single tick of ν. The second possibility is ruled out since $\nu_2 = \nu_3$. So we conclude that X is not a superkey of (R_i, μ_i). Thus, ν does not cover all non-empty ticks of μ_i. Since ν covers at least one non-empty tick of μ_i, the TFD $X \longrightarrow_\nu A$ is a partial TFD wrt μ_i. But this TFD is not a partial one wrt $\nu_3 = \nu_2$ by the construction of ν_2. Hence, the number of partial TFDs wrt ν_3 is strictly less than that wrt μ_i. Hence $n > n'$. Therefore, $(a, c, n) > (a', c', n')$ as desired.

Since the algorithm terminates, it follows from the termination condition in the algorithm that each scheme in the final decomposition is in TBCNF. We are only left to prove that the resulting set of schemes is a lossless decomposition. For this purpose, we prove that it is a tickwise-lossless decomposition and then Proposition 4 states that it is also a lossless decomposition.

Consider the central step of the algorithm: A scheme (R_i, μ_i) is decomposed into $\rho_i = \{(R_i, \nu_1), (R_i - A, \nu_2), (XA, \nu_3)\}$. First we show that such a decomposition is tickwise lossless wrt F. Consider a non-empty tick k of μ_i. By the ways that ν_1, ν_2 and ν_3 are constructed, there are two cases to be considered: (i) there exists l_1 such that $\mu_i(k) = \nu_1(l_1)$, and (ii) there exist l_2 and l_3 such that $\mu_i(k) = \nu_2(l_2)$ and $\mu_i(k) \subseteq \nu_3(l_3)$. For case (i), we have $\mathbf{MaxSub}(\mu_i(k), \rho_i) = \{(R_i, \nu_1)\}$. Since each tick of ν_1 is some tick of μ_i, it is easily seen that ρ_i is tickwise lossless for tick k. Consider case (ii). We have $\mathbf{MaxSub}(\mu_i(k), \rho_i) = \{(R_i - A, \nu_2), (XA, \nu_3)\}$ and $F \models X \longrightarrow_{\nu_3} A$. By Theorem 4, $(R_i - A, \nu_2)$ and (XA, ν_3) is a lossless decomposition of (R_i, ν_2) wrt F. Let $\mathtt{M} = (R_i, \mu_i, \phi_i)$ be a temporal module that satisfies F, and $\mathtt{M}' = (R_i, \nu_2, \phi_i')$ be the temporal module such that for each non-empty tick l of ν_2, $\phi_i'(l) = \phi_i(j)$, where $\mu_i(j) = \nu_2(l)$. Let $(R_i - A, \nu_2, \phi'') = \mathbf{Up}(\pi^T_{R_i - A}(\mathtt{M}), \nu_2)$ and $(XA, \nu_3, \phi''') = \mathbf{Up}(\pi^T_{XA}(\mathtt{M}), \nu_3)$. It is easily seen that $(R_i - A, \nu_2, \phi'') = \pi^T_{R_i - A}(\mathtt{M}')$ and $(XA, \nu_3, \phi''') = \mathbf{Up}(\pi^T_{XA}(\mathtt{M}'), \nu_3)$. Since $(R_i - A, \nu_2)$ and (XA, ν_3) form a lossless decomposition of (R_i, ν_2), $\phi_i'(l_2) = \phi''(l_2) \bowtie \phi'''(l_3)$, where l_3 is the integer such that $\nu_2(l_2) \subseteq \nu_3(l_3)$. By the definition of $\mathtt{M}'$, we have $\phi_i(k) = \phi''(l_2) \bowtie \phi'''(l_3)$. Since $\mu_i(k) = \nu_2(l_2)$ and $\mu_i(k) \subseteq \nu_3(l_3)$, it follows from the definition that ρ_i is tickwise lossless for tick k. By combining cases (i) and (ii) above, we conclude that ρ_i is a tickwise lossless decomposition of (R_i, μ_i) wrt F. We now show that the decomposition obtained by the algorithm is tickwise lossless. We do this by induction on the number of iterations of the algorithm. For notational convenience, we assume $\rho_1 = \{(R_1, \mu_1), \ldots, (R_n, \mu_n)\}$ is the decomposition before entering an iteration and (R_1, μ_1) is the scheme that is not TBCNF and is decomposed into three schemes $\rho_2 = \{(S_1, \nu_1), (S_2, \nu_2), (S_3, \nu_3)\}$, and the decomposition after the iteration is $\rho = (\rho_1 \cup \rho_2) - \{(R_1, \mu_1)\}$. As shown above, ρ_2 is always a tickwise lossless decomposition of (R_1, μ_1). We now show that after each iteration, the decomposition ρ is always tickwise lossless. As the basic step, i.e., zero iterations, it is trivial that the decomposition is tickwise lossless. Now suppose that the decomposition is tickwise lossless after $L - 1$ iterations and consider the decomposition right after the L-th iteration. By the assumptions above and the fact that the three new schemes generated at each iteration form a tickwise lossless decomposition of the scheme they replace, we only need to show that the hypothesis of Lemma 4 is true.

Consider the hypothesis (a) of Lemma 4, i.e.,

$$\mu(k) \subseteq \mu_1(k_1) \text{ implies } \mathbf{MaxSub}(\mu_1(k_1), \rho_2) = \rho_2 \cap \mathbf{MaxSub}(\mu(k), \rho)$$

Suppose $\mu(k) \subseteq \mu_1(k_1)$. Assume that (S_j, ν_j) is in $\mathbf{MaxSub}(\mu_1(k_1), \rho_2)$ for some $1 \leq j \leq 3$. We need to show that (S_j, ν_j) is also in $\mathbf{MaxSub}(\mu(k), \rho)$. Since (S_j, ν_j) is in $\mathbf{MaxSub}(\mu_1(k_1), \rho_2)$, there exists k_2 such that $\mu_1(k_1) \subseteq \nu_j(k_2)$. Since $\mu(k) \subseteq \mu_1(k_1)$, we have $\mu(k) \subseteq \mu_1(k_1) \subseteq \nu_j(k_2)$. Hence, (S_j, ν_j) is in $\mathbf{MaxSub}(\mu(k), \rho)$ as desired. Assume now (S_j, ν_j) is in $\mathbf{MaxSub}(\mu(k), \rho)$ for some $1 \leq j \leq 3$. Thus, there exists j' such that $\mu(k) \subseteq \nu_j(j')$. Since $\mu(k) \subseteq \mu_1(k_1)$, we have $\mu_1(k_1) \cap \nu_j(j') \neq \emptyset$. However, by the construction of ν_j, it is easily seen that if tick j' of ν_j overlaps with tick k_1 of μ_1, then tick j' of ν_j

covers tick k_1 of μ_1. Hence, $\mu_1(k_1) \subseteq \nu_j(j')$ and, therefore, (S_j, ν_j) is in $\mathbf{MaxSub}(\mu_1(k_1), \rho_2)$ as desired.

Consider the hypothesis (b) of Lemma 4, i.e., $\rho_2 \cap \mathbf{MaxSub}(\mu(k), \rho) \neq \emptyset$ iff $(R_1, \mu_1) \in \mathbf{MaxSub}(\mu(k), \rho_1)$. We first establish the "only-if" part. Suppose $\rho_2 \cap \mathbf{MaxSub}(\mu(k), \rho) \neq \emptyset$. Thus, there exists k' such that $\mu(k) \subseteq \nu_j(k')$ for some $1 \leq j \leq 3$. If $j = 1$ or 2, it is easily seen that $\nu_j(k') = \mu_1(k'')$ for some k'', and hence (R_1, μ_1) is in $\mathbf{MaxSub}(\mu(k), \rho_1)$. On the other hand, suppose $j = 3$, Since $\nu_3(k')$ denotes a tick of ν_3 which is a combination of some ticks in μ_1, there exist $k''_1, \ldots, k''_p$ such that $\nu_3(k') = \mu_1(k''_1, \ldots, k''_p)$. Observe that each tick of the temporal types obtained by the TBCNF decomposition algorithm is a combination of ticks of μ. Thus, each $\mu_1(k''_i)$ for $1 \leq i \leq p$ is a combination of several ticks of μ. It is then easily seen that there exists k'' such that $\mu(k) \subseteq \mu_1(k'')$ since $\mu(k) \subseteq \nu_3(k') = \mu_1(k''_1, \ldots, k''_p)$, and hence (R_1, μ_1) is in $\mathbf{MaxSub}(\mu(k), \rho_1)$. To establish the "if" part of the hypothesis (b), suppose (R_1, μ_1) is in $\mathbf{MaxSub}(\mu(k), \rho_1)$. Then there exists a tick k' of μ_1 such that $\mu(k) \subseteq \mu_1(k')$. By the construction of the types ν_1, ν_2 and ν_3, it is easily seen that there exist k'' and $1 \leq j \leq 3$ such that $\mu_1(k') \subseteq \nu_j(k'')$, and hence $\mu(k) \subseteq \nu_j(k'')$. Clearly, the last statement implies that there exists a scheme in ρ_2 such that the scheme is in $\mathbf{MaxSub}(\mu(k), \rho)$.

Thus, we have established that the two hypotheses of Lemma 4 hold for the decomposition at the L-th iteration. Thus, it follows from the induction hypothesis that ρ is tickwise lossless decomposition of (R, μ) wrt F since ρ_1 is tickwise lossless decomposition of (R, μ) wrt F. The induction proof therefore establishes the fact that the decomposition obtained by the algorithm is a tickwise lossless decomposition of (R, μ) wrt F. By Proposition 4, the decomposition is a lossless decomposition of (R, μ) wrt F. This concludes our proof. $\square$

A.3 Proof of Theorem 6

Lemma 5 Let F be a set of TFDs, (R, μ) a scheme, $X \subseteq R$ a set of attributes, $B \in R$ an attribute, μ_r a subtype of μ, and
$F_{\mu_r} = \{X \longrightarrow_\nu A \mid X \longrightarrow_\nu A \in \mathbf{MinCov}(F) \text{ and } \exists j, i \ \mu_r(j) \neq \emptyset \ \wedge \ \mu_r(j) \subseteq \nu(i)\}$.
Then, the following holds:

$$(B, \mu_r) \in \overline{X}^+ \text{ wrt } F \quad \text{ implies } \quad (B, \mu_r) \in \overline{X}^+ \text{ wrt } F_{\mu_r}$$

Proof. Since (B, μ_r) is derived by the algorithm in Figure 3 at a certain step j, we know that $\mu_r = glb(\mu_1, \ldots, \mu_k, \mu')$ where μ' is the type of the TFD in F considered by the algorithm at step j and having B among the attributes on the right side, while the $\mu_1, \ldots, \mu_k$ are types present in the pairs $(A_i, \mu_i) \in X^{(j-1)}$ needed for the application of that TFD. However, considering step 2 of the algorithm, it is clear that $\mu_1, \ldots, \mu_k$ must be either μ_{Top} or glbs of types appearing in TFDs considered by the algorithm in previous steps. The lattice of types insures that, for arbitrary types $\lambda_1, \lambda_2, \lambda_3$, $glb(\lambda_1, glb(\lambda_2, \lambda_3)) = glb(\lambda_1, \lambda_2, \lambda_3)$. Hence, we know that $\mu_r = glb(\nu_1, \ldots, \nu_n)$ where $\nu_1, \ldots, \nu_n$ are all the types appearing in the TFDs of F used by the algorithm to derive (B, μ_r). We show that all these TFDs are not only in F, but also in F_{μ_r}. Suppose $V \longrightarrow_\nu A \in F$ but $V \longrightarrow_\nu A \notin F_{\mu_r}$. Then, from the definition of F_{μ_r}, we know that $\forall i, j \quad \mu_r(i) \neq \emptyset \Rightarrow \mu_r(i) \not\subseteq \nu(j)$. This means that $\nu \notin \{\nu_1, \ldots, \nu_n\}$ since, by definition of glb, every tick of μ_r should be covered by a tick of each one of the $\nu_1, \ldots, \nu_n$. We can conclude that only TFDs in F_{μ_r} are useful in the algorithm to obtain (B, μ_r) and this is equivalent to say that if (B, μ_r) is obtained by the algorithm, it can be obtained considering indifferently F_{μ_r} or F. □

Proof. The proof of termination is trivial since step 1 and step 2 are iterations bounded by the number of TFDs in $\mathbf{MinCov}(F)$. Step 3 is an iteration bounded by the number types in P, and step 4 is limited by the length of the decomposition resulting by previous steps.

We now prove that the decomposition $\rho = \{(R_1, \lambda_1), \ldots, (R_k, \lambda_k)\}$ obtained from the algorithm wrt (R, μ) and F preserves dependencies. Suppose $\mathtt{M} = (R, \mu, \phi)$ does not satisfy F. By the definition of the dependency preservation, we only need to show that $\mathbf{Up}(\pi^T_{R_i}(\mathtt{M}), \lambda_i)$ does not satisfy $\pi_{R_i}(F)$ for some $1 \leq i \leq k$. Since $\mathtt{M}$ does not satisfy F, there must exist a TFD $X \longrightarrow_\nu A$ in $\mathbf{MinCov}(F)$ that is not satisfied by $\mathtt{M}$. That is, there exist tuples t_1 and t_2 and ticks i_1 and i_2 of μ such that $t_1[X] = t_2[X]$, t_1 is in $\phi(i_1)$, t_2 is in $\phi(i_2)$, and there exists j such that $\mu(i_1, i_2) \subseteq \nu(j)$, but $t_1[A] \neq t_2[A]$. Suppose P is the partition of μ created by the T3NF decomposition algorithm. Then there exist μ_{r1} and μ_{r2} in P such that $\mu(i_1) = \mu_{r1}(l_1)$ and $\mu(i_2) = \mu_{r2}(l_2)$ for some l_1 and l_2, and hence $X \longrightarrow_\nu A$ is in both $F_{\mu_{r1}}$ and $F_{\mu_{r2}}$ by the definition in the algorithm. By the algorithm and the fact that $X \rightarrow A$ is in the $\pi_\emptyset(\mathbf{MinCov}(F))$ set, the scheme (S, λ) is in ρ with $XA \subseteq S$ and $\lambda = cop(\mu_r \cup \ldots \cup \mu_s, F_{X \rightarrow A})$, where $\mu_r, \ldots, \mu_s$ and $F_{X \rightarrow A}$ are as defined in the algorithm. Since $X \longrightarrow_\nu A$ is in $F_{\mu_{r1}}$ and in $F_{\mu_{r2}}$, it follows that μ_{r1} and μ_{r2} must be among $\mu_r, \ldots, \mu_s$ and $X \longrightarrow_\nu A$ must be in $F_{X \rightarrow A}$. It suffices now to show that $\mathbf{Up}(\pi^T_S(\mathtt{M}), \lambda)$ does not satisfy $X \longrightarrow_\nu A$. Let $\mu_\cup = \mu_r \cup \ldots \cup \mu_s$ and $\mathtt{M}_\cup = \mathbf{Up}(\pi^T_S(\mathtt{M}), \mu_\cup)$. We have the equation:

$$\mathbf{Up}(\pi^T_S(\mathtt{M}), \lambda) = \mathbf{Up}(\mathtt{M}_\cup, \lambda).$$

This equation follows immediately from the following observations: (1) a tick of μ is covered by a tick of λ iff it is covered by a tick of $\mu_\cup$, and (2) since $\mu_\cup$ is a subtype of μ, the effect of the $\mathbf{Up}$ operation in defining $\mathtt{M}_\cup$ is only to drop the values of the windowing function for the ticks not in $\mu_\cup$, hence not covered by any tick of λ. By Proposition 5 and the above equation, to show that $\mathbf{Up}(\pi_S(\mathtt{M}), \lambda)$ does not satisfy $X \longrightarrow_\nu A$, we only need to show that $\mathtt{M}_\cup$ does not satisfy $X \longrightarrow_\nu A$. Since μ_{r_1} and μ_{r_2} are among $\mu_r, \ldots, \mu_s$, there exist l'_1 and l'_2 such that $\mu(i_1) = \mu_{r1}(l_1) = \mu_\cup(l'_1)$ and $\mu(i_2) = \mu_{r2}(l_2) = \mu_\cup(l'_2)$. Let $\mathtt{M}_\cup = (S, \mu_\cup, \phi_\cup)$. By the definitions of the projection and **Up** operations and the fact that $\mu_\cup$ is a subtype of μ, it is easily seen that $t_1[S]$ is in $\phi_\cup(l'_1)$ and $t_2[S]$ is in $\phi_\cup(l'_2)$ since t_1 is in $\phi(i_1)$ and t_2 is in $\phi(i_2)$. Since $\mu(i_1, i_2) \subseteq \nu(j)$, it follows that $\mu_\cup(l'_1, l'_2) \subseteq \nu(j)$. Since $t_1[X] = t_2[X]$ but $t_1[A] \neq t_2[A]$, and also $XA \subseteq S$, it then follows that $\mathtt{M}_\cup$ does not satisfy $X \longrightarrow_\nu A$. This concludes our proof that ρ preserves dependencies.

We now show that each scheme in ρ is in T3NF. For each scheme (R_i, λ_i) in ρ, two cases arise: (1) $R_i = Z$, where Z is a candidate key of (R, λ_i) and λ_i is one of the types in the partition P, and (2) $R_i = XA$, where $X \rightarrow A$ is in $\pi_\emptyset(\mathbf{MinCov}(F))$ and λ_i is computed by the function *cop*() with the arguments as explained in the algorithm. In case (1), we know that Z is also a key of (Z, λ_i). Indeed, if there exists a proper subset Y of Z such that $Y \longrightarrow_{\lambda_i} Z$, then $Y \longrightarrow_{\lambda_i} R$, and this contradicts the fact that Z is a candidate key of (R, λ_i). Therefore, for each TFD $V \longrightarrow_\nu W$, with $VW \subseteq Z$, we know that W must consist of prime attributes. By definition, (Z, λ_i) is in T3NF. In case (2) we consider a generic scheme (XA, λ') with $\lambda' = cop(\lambda, F_{X\rightarrow A})$, where $\lambda = \mu_r \cup \cdots \cup \mu_s$ and $F_{X \rightarrow A}$ are as defined in the algorithm. By the algorithm, for each μ_i $(r \leq i \leq s)$, there exists a TFD $X\nu_j A$ in $F_{X\rightarrow A}$ such that a tick of μ_i is covered by some tick of ν_j. Since μ_i is a partition of μ from the algorithm, it is easily seen that each tick of μ_i is covered by some tick of ν_j (otherwise, μ_i will be partitioned further). Hence, each tick of λ is covered by some tick of ν_j. Therefore, it is easily seen that $X \longrightarrow_\lambda A$ is logically implied by F since each $X \longrightarrow_{\nu_j} A$ in $F_{X\rightarrow A}$ is logically implied by F. We now claim that X is a candidate key of (XA, λ'). Indeed, we have shown that $X \longrightarrow_\lambda A$ is logically implied by F. By the definition of *cop*(), it is easily seen that $X \longrightarrow_{\lambda'} A$ is logically implied by F. Now suppose there is a proper subset V of X such that $V \longrightarrow_{\lambda'} XA$, hence $V \longrightarrow_{\lambda'} A$ is logically implied by F and then, $V \rightarrow A$ is logically implied by $\pi_\emptyset(F)$. Since $X \rightarrow A$ is in $\pi_\emptyset(\mathbf{MinCov}(F))$ and, by the construction of $\mathbf{MinCov}(F)$, $\pi_\emptyset(\mathbf{MinCov}(F))$ is the same as the minimal cover of $\pi_\emptyset(F)$, it follows that $X \rightarrow A$ is in the minimal cover of $\pi_\emptyset(F)$ and $V \rightarrow A$ is logically implied by $\pi_\emptyset(F)$. This is a contradiction since we may replace $X \rightarrow A$ by $V \rightarrow A$ in the minimal cover of $\pi_\emptyset(F)$.

Let us now consider an arbitrary TFD $V \longrightarrow_\nu B$ that is logically implied by F such that $VB \subseteq XA$, $B \notin V$ and there exists a tick of λ' which is covered by some tick of ν. If $B \in X$, then B is prime since X is a candidate key of (XA, λ'). Thus, we need only consider the case when B is not in X, i.e., $B = A$ and $V \longrightarrow_\nu A$ is logically implied by F. Since $A = B \notin V$, it follows that V is a subset of X. We know that V cannot be a proper subset of X since otherwise, as shown earlier, $X \rightarrow A$ cannot be in $\pi_\emptyset(\mathbf{MinCov}(F))$. Therefore, $V = X$ and we need only consider the TFD $X \longrightarrow_\nu A$ that is logically implied by F with the assumption that there exists at least one tick of λ' that is covered by a tick of ν. We now show that the second condition of T3NF is satisfied by $X \longrightarrow_{\nu} A$. As shown above, X is a candidate key for (XA, λ'). Assume now that two non-empty ticks $\lambda'(i_1)$ and $\lambda'(i_2)$, where $i_1 \neq i_2$, are covered by a single tick j of ν. Let $\mathcal{V} = \{\nu' | X \longrightarrow_{\nu'} A \in \mathbf{MinCov}(F)\}$. By the definition of minimal cover, we know $\nu \preceq_C \mathcal{V}$. Thus, there exist ν' in $\mathcal{V}$ and j' such that $\nu(j) \subseteq \nu'(j')$. Since $\lambda'(i_1, i_2) \subseteq \nu(j) \subseteq \nu'(j')$ and λ' is obtained by collapsing ticks of $\lambda = \mu_r \cup \cdots \cup \mu_s$, without loss of generality, we may assume that $\emptyset \neq \mu_r(i'_1) \subseteq \lambda'(i_1)$ for some i'_1. Thus, $\mu_r(i'_1) \subseteq \nu'(j')$. By the definition of F_{μ_r}, we know that $X \longrightarrow_{\nu'} A$ is in F_{μ_r}. Therefore, by definition in the algorithm again, we know $X \longrightarrow_{\nu'} A$ is in $F_{X \to A}$. Thus, $\lambda'(i_1, i_2) \subseteq \nu'(j')$. By the definition of *cop*() function, it follows that there exist a TFD $X \longrightarrow_{\nu_i} A$, in $F_{X \to A}$, that is logically implied by F and integers k' and i_3 with $i_3 \neq i_1$ such that $\lambda'(i_1)$ and $\lambda'(i_3)$ are contained in $\nu_i(k')$ and $\lambda(i_3)$ is not contained in $\nu'(j')$. Since $\nu(j) \subseteq \nu'(j')$ and $\nu'(j')$ does not contain $\lambda'(i_3)$, it is clear that $\nu(j)$ does not contain $\lambda'(i_3)$. This is exactly required by the second condition of T3NF. Hence, (XA, λ) is in T3NF.

Finally, we prove that the decomposition ρ is lossless. By Proposition 4, it suffices to show that ρ is tickwise lossless. Let us consider a generic nonempty tick k of μ and assume $\mathbf{MaxSub}(\mu(k), \rho) = \{(R_1, \lambda_1), \ldots, (R_m, \lambda_m)\}$. We only need to prove that for each module $\mathtt{M} = (R, \mu, \phi)$ and corresponding projections $\mathtt{M}_i = (R_i, \mu, \phi_i) = \mathbf{Down}(\mathbf{Up}(\pi^T_{R_i}(\mathtt{M}), \lambda_i), \mu)$ for $i = 1, \ldots, m$, the following holds:

$$\phi(k) = \phi_1(k_1) \bowtie \cdots \bowtie \phi_m(k_m)$$

where $\mu(k) \subseteq \lambda_i(k_i)$ for each $1 \leq i \leq m$. By the definitions of **Up** and **Down** operations and the fact that $\mu(k) \subseteq \lambda_i(k_i)$ for each $1 \leq i \leq m$, it is easily seen that $\phi_1(k_1) \bowtie \cdots \bowtie \phi_m(k_m) \subseteq \phi(k)$. We only need to show that $\phi(k) \subseteq \phi_1(k_1) \bowtie \cdots \bowtie \phi_m(k_m)$. Suppose by contradiction that there exists a tuple t in $\phi_1(k_1) \bowtie \ldots \bowtie \phi_m(k_m)$ that is not in $\phi(k)$. This tuple results from a natural join of tuples, hence for each $1 \leq i \leq m$, there exist $t_i \in \phi_i(k_i)$ such that $t[R_i] = t_i$. For each $1 \leq i \leq m$, since $\phi_i(k_i) = \bigcup_{j:\mu(j) \subseteq \lambda(k_i)} \phi(j)$ by definition, we know that there exists a tuple $t_i^o \in \phi(k_i^o)$ for some k_i^o such that $\mu(k, k_i^o) \subseteq \lambda_i(l_i)$ for some l_i and $t_i^o[R_i] = t_i$. This intuitively says that tuples t_i are the projection of some tuple given in the original module at tick k_i^o where k_i^o and k are covered by the same tick $k_i = l_i$ of λ_i. Let μ_k be in P and such that there exists a tick of μ_k that equals the tick k of μ. We know there exists a scheme (V, μ_l) in $\mathbf{MaxSub}(\mu(k), \rho)$ such that V contains a temporal candidate key for (R, μ_k) and μ_k is a subtype of μ_l. [Indeed, from the algorithm, either a scheme (V, μ_l), with V containing a temporal candidate key of (R, μ_k) and μ_k a subtype of μ_l, is already in ρ, or (Z, μ_k) is added by step 3, where Z is a candidate key of (R, μ_k). In the first case, (V, μ_r) is in $\mathbf{MaxSub}(\mu(k), \rho)$ since a tick of μ_k covers tick k of $\mu(k)$ by our choice of μ_k and μ_k is a subtype of μ_l. In the second case, (Z, μ_k) is in

MaxSub$(\mu(k), \rho)$ since $\mu(k)$ is covered by a tick of μ_k by our choice of μ_k.] Without loss of generality, assume (R_1, λ_1) has the property that (a) (R_1, λ_1) is in **MaxSub**$(\mu(k), \rho)$, (b) R_1 contains a candidate key of (R, μ_k), and (c) μ_k is a subtype of λ_1. By the definitions of **Up** and **Down** and the fact that a tick of λ_1 is exactly the tick k of μ, $k_1^o = k$. Since $t_1 = t_1^o[R_1]$ and $t[R_1] = t_1$, we have $t[R_1] = t_1^o[R_1]$. Note that t_1^o is in $\phi(k_1^o) = \phi(k)$ since $k_1^o = k$ This means that there exists a tuple $t_1^o \in \phi(k)$ such that $t_1^o[R_1] = t[R_1]$. Now we know that $t \neq t_1^o$ since we assume t is not in $\phi(k)$. Thus, there exists A such that $t[A] \neq t_1^o[A]$. Since R_1 contains a candidate key of (R, μ_k) wrt F, it contains a candidate key of (R, μ_k) wrt F_{μ_k} by Lemma 5. Thus, every attribute of R, and hence A, appears in $\overline{Z}^+$, where $Z \subseteq R_1$ is a temporal candidate key of (R, μ_k). Consider now the algorithm in Figure 3 that computes $\overline{Z}^+$ wrt F_{μ_k}. At each step the algorithm takes a TFD $X_i \longrightarrow_{\nu_i} A_i$ from F_{μ_k} and adds a pair $(A_i, \overline{\nu}_i)$ where either A_i is an attribute not appearing in the previously computed set or $\overline{\nu}_i$ is a new type. Without loss of generality, suppose A is the first attribute added by the algorithm such that $t_1^o[A] \neq t[A]$ and $Z, A_1, \ldots, A_s$ the attributes added before A and $t[ZA_1 \ldots A_s] = t_1^o[ZA_1 \ldots A_s]$. Then we know that there exists a TFD $X \longrightarrow_{\nu} A$ in F_{μ_k} such that $X \subseteq ZA_1 \ldots A_s$.

Since we know that $X \longrightarrow_{\nu} A$ is in F_{μ_k}, we know that $X \rightarrow A$ is in $\pi_{\emptyset}(\mathbf{MinCov}(F))$. Therefore, a scheme (XA, λ) is added to ρ at Step 2 of the algorithm. This scheme can only be taken out from ρ at Step 4, if a scheme (R_i, λ) is in ρ with $XA \subseteq R_i$. Without loss of generality, assume this scheme is (R_2, λ_2), i.e., $XA \subseteq R_2$ and $\lambda_2 = \lambda = \mathit{cop}(\mu_r \cup \ldots \cup \mu_s, F_{X \rightarrow A})$ where $\{\mu_r, \ldots, \mu_s\} = \{\mu_j \in P \mid \exists \nu_i : X \longrightarrow_{\nu_i} A \in F_{\mu_j}\}$ and $F_{X \rightarrow A} = \{X \longrightarrow_{\nu_j} A \mid \exists \mu_j \in P : X \longrightarrow_{\nu_i} A \in F_{\mu_j}\}$. In particular the following holds:

- $\mu_k \preceq \lambda_2$.

Indeed, since $X \longrightarrow_{\nu} A$ is in F_{μ_k}, μ_k is among $\mu_r, \ldots, \mu_s$. Now it is easily seen that $\mu_k \preceq \mu_r \cup \ldots \cup \mu_s$. By the definition of function $\mathit{cop}()$, $\mu_r \cup \ldots \cup \mu_s \preceq \lambda_2$. Hence, $\mu_k \preceq \lambda_2$. Note also that the scheme (R_2, λ_2) is in **MaxSub**$(\mu(k), \rho)$ by the fact that $\mu(k)$ is covered by a tick of μ_k and $\mu_k \preceq \lambda_2$.

We know that $t_1^o[X] = t[X]$ since $X \subseteq ZA_1 \ldots A_s$. Since $t[R_2] = t_2$ and $X \subseteq R_2$, we have $t_1^o[X] = t[X] = t_2[X]$. Note that $t_2 = t_2^o[R_2]$, where t_2^o is in $\phi(k_2^o)$ and k_2^o and k are both covered by the tick k_2 of λ_2. We now know that (i) $t_2^o[X] = t_2[X] = t_1^o[X]$ since $X \subseteq R$, and (ii) t_1^o is in $\phi(k)$ and t_2^o is in $\phi(k_2^o)$ and $\mu(k, k_2^o) \subseteq \lambda_2(l_2)$ for some l_2. Let $\mathcal{V} = \{\nu_r \mid X \longrightarrow_{\nu_r} A \in \mathbf{MinCov}(F)\}$. By the definition of $\mathbf{MinCov}(F)$, we know that $\lambda_2 \preceq_C \mathcal{V}$. Thus, there exists ν_r in $\mathcal{V}$ such that $\lambda_2(k_2) \subseteq \nu_r(l')$ for some l'. Since $\mu(k, k_2^o) \subseteq \lambda_2(k_2) \subseteq \nu_r(l')$, $X \longrightarrow_{\nu_r} A$ is in F_{μ_r}, we know that $t_1^o[A] = t_2^o[A]$ by the assumption that `M` satisfies F and the facts (i) and (ii) above. Therefore, $t_1^o[A] = t_2^o[A] = t[A]$ since $A \in R_2$ and $t[R_2] = t_2 = t_2^o[R_2]$. This is a contradiction since we assume $t_1^o[A] \neq t[A]$. □