Temporal Dependencies with Order Constraints

Jef Wijsen

December 20, 1999

TR-47

A TimeCenter Technical Report

Title	Temporal Dependencies with Order Constraints
Author(s)	Jef Wijsen
Publication History	This manuscript has been conditionally accepted for publication in *IEEE Trans. on Knowledge and Data Engineering*. December 1999. A TIMECENTER Technical Report.

TIMECENTER Participants

Aalborg University, Denmark
Christian S. Jensen (codirector), Michael H. Böhlen, Heidi Gregersen, Dieter Pfoser, Simonas Šaltenis, Janne Skyt, Giedrius Slivinskas, Kristian Torp

University of Arizona, USA
Richard T. Snodgrass (codirector), Bongki Moon

Individual participants
Curtis E. Dyreson, Bond University, Australia
Fabio Grandi, University of Bologna, Italy
Nick Kline, Microsoft, USA
Gerhard Knolmayer, Universty of Bern, Switzerland
Thomas Myrach, Universty of Bern, Switzerland
Kwang W. Nam, Chungbuk National University, Korea
Mario A. Nascimento, University of Alberta, Canada
John F. Roddick, University of South Australia, Australia
Keun H. Ryu, Chungbuk National University, Korea
Michael D. Soo, amazon.com, USA
Andreas Steiner, TimeConsult, Switzerland
Vassilis Tsotras, University of California, Riverside, USA
Jef Wijsen, University of Mons-Hainaut, Belgium
Carlo Zaniolo, University of California, Los Angeles, USA

For additional information, see The TIMECENTER Homepage:
URL: `<http://www.cs.auc.dk/TimeCenter>`

The TIMECENTER icon on the cover combines two "arrows." These "arrows" are letters in the so-called *Rune* alphabet used one millennium ago by the Vikings, as well as by their precedessors and successors. The Rune alphabet (second phase) has 16 letters, all of which have angular shapes and lack horizontal lines because the primary storage medium was wood. Runes may also be found on jewelry, tools, and weapons and were perceived by many as having magic, hidden powers.

The two Rune arrows in the icon denote "T" and "C," respectively.

Abstract

We propose a temporal dependency, called *trend dependency* (TD), which captures a significant family of data evolution regularities. A simple example of such regularity might be: *"Salaries of employees do not decrease."* TDs compare attributes over time using operators of $\{<, =, >, \leq, \geq, \neq\}$. We define a satisfiability problem that is the dual of the logical implication problem for TDs, and we investigate the computational complexity of both problems. We provide an axiomatization of logical implication. As TDs allow expressing meaningful trends, "mining" them from existing databases is interesting. For the purpose of TD mining, TD satisfaction is characterized by the common notions of *support* and *confidence*. We study the problem TDMINE: Given a temporal database, mine the TDs that conform to a given template and whose support and confidence exceed certain threshold values. The complexity of TDMINE is studied, as well as algorithms to solve it. A comparison with related work is provided at the end of the paper. We show that TDs can express several temporal dependencies found in the literature.

1 Introduction

Lately, there has been a growing research interest in temporal database integrity. Temporal constraints can take different forms. They have been expressed using first-order temporal logic (FOTL) [8, 9]. Alternatively, one can study restricted classes of FOTL formulas, which may be called *temporal dependencies*. A comprehensive overview of temporal dependencies has been given by Jensen et al. [20]. In this paper, we introduce a new temporal dependency, called *trend dependency* (TD), which captures a significant class of data evolution constraints. Two examples of such constraints, taken from recent work, are *"Salaries of employees should never decrease"* [8] and *"A faculty's rank cannot change during an academic year"* [29].

Until now, *database integrity* has been the single most important motivation for studying temporal dependencies. Temporal dependencies allow capturing more real-world knowledge in a database schema by placing restrictions on how the data can change over time. In this paper, we also look at TDs from a *knowledge discovery* (or *data mining*) point of view: As TDs allow expressing significant real-world knowledge, discovering them from existing databases is interesting and important.

We briefly explain the type of logic formulas expressed by TDs. In this study, time is represented by the set of natural numbers $\mathbb{N}$ $(= \{1, 2, 3, \ldots\})$. A temporal relation is viewed as a time series $I = \langle I_1, I_2, I_3, \ldots\rangle$ of conventional "snapshot" relations, all over the same set of attributes. Intuitively, one may think of I_i as the family of tuples valid at time i. For example, consider the relation schema $\{\texttt{SS\#}, \texttt{Rank}, \texttt{Sal}\}$. A tuple $\{\texttt{SS\#} : x, \texttt{Rank} : y, \texttt{Sal} : z\}$ of I_i means that at time i, the employee with social security number x has rank y and salary z. Employees are uniquely identified by their social security number.

Checking temporal constraints typically involves comparing tuples that are valid at different points of time. For example, checking *"Salaries of employees should never decrease,"* requires comparing employee records at time i with records at the next time $i + 1$, for each time point i. To relate time points in a temporal constraint, we make use of binary relations on the set of time points, called *time accessibility relations* (TARs). The TAR emerging in the running example is $\{(i, i+1) \mid i \in \mathbb{N}\}$, which is called *Next*. At this point, the meaning of TDs can be most easily explained by expressing them in a tuple-oriented relational calculus. We use the predicate $emp(s, i)$ with the meaning that tuple s belongs to I_i. The constraint under consideration can then be expressed as:

$$\begin{aligned}\forall i \forall j \forall s \forall t \{ \quad [\quad & emp(s,i) \wedge emp(t,j) \wedge \\ & Next(i,j) \wedge \\ & s(\texttt{SS\#}) = t(\texttt{SS\#}) \ ] \rightarrow s(\texttt{Sal}) \leq t(\texttt{Sal}) \ \}\end{aligned}$$

where $Next(i, j)$ means that (i, j) belongs to *Next*. We will denote this constraint as

$$(\texttt{SS\#}, =) \rightarrow_{Next} (\texttt{Sal}, \leq)$$

and call it a *trend dependency* (TD). TDs generalize functional dependencies (FDs) in two ways: first, by comparing tuples over time, and second, by comparing attributes with any operator of $\{<, =, >, \leq, \geq, \neq\}$.

TDs can capture several temporal dependencies found in the literature. They seem to be among the first temporal dependencies that compare attributes using operators other than equality. In this paper, we are going to address some important practical problems that apply to any new type of dependency:

Logical implication problem Given a set Σ of TDs and a single TD σ, if a temporal relation satisfies Σ, does it necessarily satisfy σ as well?

Satisfiability problem Can a specified set Σ of TDs be satisfied in a "non-trivial" way?

The satisfiability problem starts from a specified family Σ of TDs and looks for a particular temporal relation satisfying Σ. The following data mining problem is in some respect the inverse of the satisfiability problem. It starts from a given temporal relation and looks for TDs satisfied by it.

TD mining problem Given a temporal relation, which TDs are satisfied "to a high degree"?

At this point, it is not possible to give a precise characterization of the latter problem, but the idea can be illustrated as follows. Imagine a temporal relation containing salary histories of employees. Assume that no integrity constraints have been specified concerning the evolution of salaries. Inspecting the data, one may observe employees with increasing as well as decreasing salaries. The task then is to find out which trend is the "stronger" one: Do salaries of employees generally increase or decrease? To answer such question, we need a measure for characterizing the "strength" of a trend. This will be done by adapting the notions of *confidence* and *support*, which are commonly used in the field of data mining (see for example [3]).

The outline of the paper is as follows. Section 2 introduces two components of TDs, namely *directed attribute sets* (DASs) for comparing tuples, and *time accessibility relations* (TARs) to relate time points. Having defined these components, we can formalize the notion of TD in Section 3. Section 4 concerns the complexity of the logical implication problem and the satisfiability problem. An axiomatization of logical implication is provided in Section 5. Section 6 concerns mining TDs from temporal databases. The problem TDMINE is defined and its complexity is studied. TDs encompass several previous proposals of temporal dependencies found in the literature. A comparison with related work is contained in Section 7. Finally, Section 8 summarizes the most important results.

2 Preliminaries

2.1 Comparing Tuples

A basic assumption in our theoretical framework is that attributes take their values from totally ordered domains. Examples of such attributes are numerous (e.g., `Rank` and `Sal`). We note that an attribute domain can be totally ordered, even though this order does not naturally arise in integrity constraints. `SS#` can serve as an example. Administrative procedures typically rely on a particular total order on social security numbers to locate and list employee records. Nevertheless, it is likely that in practical database constraints, `SS#`-values are only compared for equality ($=$) and inequality ($\neq$). Attributes like `SS#` fit in our framework as there is no problem in using only equality and inequality constraints for certain attributes.

The following definition introduces a convenient syntactic shorthand for comparing tuples. For example, let s and t be "employee" tuples. The formula $s(\texttt{SS\#}) = t(\texttt{SS\#}) \wedge s(\texttt{Rank}) \leq t(\texttt{Rank})$ will be denoted $\Psi^*(s,t)$ where Ψ is equal to the set $\{(\texttt{SS\#},=),(\texttt{Rank},\leq)\}$ and called a DAS.

Definition 1 We assume the existence of a totally ordered, infinite set $(\mathbf{dom},\leq)$ of *constants*. We introduce two special operators $\perp$ and $\top$ as follows: For every $d_1, d_2 \in \mathbf{dom}$, $d_1 \perp d_2$ is false and $d_1 \top d_2$ is true. We assume the existence of a set $\mathbf{att}$ of *attributes*. Let $U \subseteq \mathbf{att}$. A *tuple* over U is a total function from U to $\mathbf{dom}$. If t is a tuple over U and $X \subseteq U$, then $t[X]$ denotes the tuple over X obtained by restricting the function t to X.

As the constraints considered in this paper never induce comparing distinct attributes, a single domain can be assumed without loss of generality.

Operator $\theta \in \mathbf{OP}$	Syntactic Shorthand	Inverse $\overline{\theta}$	Reciprocal $\widehat{\theta}$
$\{\}$	$\bot$	$\top$	$\bot$
$\{<\}$	$<$	$\geq$	$>$
$\{=\}$	$=$	$\neq$	$=$
$\{>\}$	$>$	$\leq$	$<$
$\{<,=\}$	$\leq$	$>$	$\geq$
$\{=,>\}$	$\geq$	$<$	$\leq$
$\{<,>\}$	$\neq$	$=$	$\neq$
$\{<,=,>\}$	$\top$	$\bot$	$\top$

Figure 1: Operators of **OP** with shorthand and inverse.

The set **OP** is defined as the powerset of $\{<,=,>\}$. That is, $\mathbf{OP} = \wp(\{<,=,>\})$. Elements of **OP** are called *operators*. The Greek lowercase letter θ will be used to denote operators. We introduce convenient syntactic shorthands for the elements of **OP**, as shown in Figure 1.

Let $\theta \in \mathbf{OP}$. The *inverse* of θ, denoted $\overline{\theta}$, is equal to $\{<,=,>\} \setminus \theta$. The *reciprocal* of θ, denoted $\widehat{\theta}$, is equal to the operator obtained from θ by interchanging $<$ and $>$.

Let U be a set of attributes (i.e., $U \subseteq \mathbf{att}$). A *directed attribute set* (DAS) over U (or simply DAS, if U is understood) is a total function from U to **OP**. The Greek uppercase letters $\Phi, \Psi, \Upsilon, \Omega$ will be used to denote DASs.

Let Φ be a DAS. The domain of Φ is denoted $atts(\Phi)$. That is, $atts(\{(A_1,\theta_1),\ldots,(A_n,\theta_n)\}) = \{A_1,\ldots,A_n\}$. The empty DAS is denoted $\emptyset$. The *reciprocal* of Φ, denoted $\widehat{\Phi}$, is the DAS over $atts(\Phi)$ such that for every $A \in atts(\Phi)$, $\widehat{\Phi}(A) = \widehat{\theta}$ where $\theta = \Phi(A)$. Φ is called *reflexive* iff for every $A \in atts(\Phi)$, $\Phi(A)$ contains the equality $(=)$ operator.

$\{(A_1,\theta_1),(A_2,\theta_2),\ldots,(A_n,\theta_n)\}$ is usually denoted $(A_1,\theta_1)(A_2,\theta_2)\ldots(A_n,\theta_n)$.

Let s,t be tuples over U. Let Φ be a DAS over some subset X of U. We say that the *tuple pair* (s,t) *satisfies* Φ, denoted $\Phi^*(s,t)$, iff for each $A \in X$, $s(A)\ \theta_A\ t(A)$ where θ_A is the shorthand for $\Phi(A)$.[1]

Let Φ, Ψ be DASs. We write $\Phi \wedge \Psi$ for the DAS Υ over $atts(\Phi) \cup atts(\Psi)$ satisfying the following conditions:

- For every $A \in atts(\Phi) \cap atts(\Psi)$, $\Upsilon(A) = \Phi(A) \cap \Psi(A)$,
- for every $A \in atts(\Phi) \setminus atts(\Psi)$, $\Upsilon(A) = \Phi(A)$, and
- for every $A \in atts(\Psi) \setminus atts(\Phi)$, $\Upsilon(A) = \Psi(A)$.

□

Example 1 Let Φ be the DAS $(\texttt{SS\#},=)(\texttt{Rank},\leq)(\texttt{Sal},<)$. The reciprocal of Φ is equal to $\widehat{\Phi} = (\texttt{SS\#},=)(\texttt{Rank},\geq)(\texttt{Sal},>)$. Φ is not reflexive as $\Phi(\texttt{Sal})$ does not contain $=$. We have $\Phi \wedge (\texttt{Rank},\neq)(\texttt{Sal},\top) = (\texttt{SS\#},=)(\texttt{Rank},<)(\texttt{Sal},<)$. □

It can be easily proved that "$\wedge$" is commutative, associative, and idempotent. The following lemma states some properties that will be used later on.

[1] Note the double use of the symbols $\bot, <, =, >, \leq, \geq, \neq, \top$: First, as a shorthand for operators of **OP**, and second, to compare elements of **dom**. This little abuse of notation does not result in any confusion, however.

Lemma 1 *Let U be a set of attributes. Let s, t be tuples over U. Let Φ, Ψ be DASs with $atts(\Phi), atts(\Psi) \subseteq U$. Then*

1. *$\emptyset^*(s,t)$ (recall $\emptyset$ is the empty DAS).*
2. *$(\Phi \wedge \Psi)^*(s,t)$ iff $\Phi^*(s,t)$ and $\Psi^*(s,t)$.*
3. *Φ is reflexive iff $\Phi^*(s,s)$.*

PROOF. Straightforward. □

2.2 Relating Time Points

In this subsection, we first define TARs and then we show how TARs can be used to model time granularities.

2.2.1 Time Accessibility Relations (TARs)

In this work, the time line is represented by the set of natural numbers $\mathbb{N}$ ($= \{1, 2, 3, \ldots\}$). Trends typically induce comparing attributes of one tuple s with attributes of another tuple t, where s and t may belong to different "snapshots." We now introduce the concept of TAR to indicate which tuples have to be compared with one another.

Definition 2 We define:

$$\mathit{Future} = \{(i,j) \mid i, j \in \mathbb{N} \text{ and } i \leq j\}.$$

Any computable subset of *Future* is called a *time accessibility relation* (TAR).[2] *Current*, *Next*, and *NextOne* are special TARs which will be frequently used in the technical treatment later:

$\mathit{Current} = \{(i,i) \mid i \in \mathbb{N}\}$,
$\mathit{Next} = \{(i,i+1) \mid i \in \mathbb{N}\}$, and
$\mathit{NextOne} = \{(1,2)\}$.

Greek lowercase letters α, β will be used to denote TARs. The *complement* of a TAR α, denoted $\neg\alpha$, is given by: $\neg\alpha = \mathit{Future} \setminus \alpha$. □

Note that $\{\}$ and *Future* itself are TARs. We emphasize that whenever the pair (i,j) belongs to a TAR α, then $i \leq j$. This restriction simplifies the technical treatment later on, without decreasing the expressiveness of TDs. It can be easily seen that *Future* is recursively enumerable. The requirement that TARs are computable means that, given a TAR α and some member (i,j) of *Future*, one is able to tell whether or not (i,j) belongs to α. TARs need not be finite. Of course, real systems can only deal with TARs that have a finite representation. The representation of TARs will be discussed in Section 4.

2.2.2 Chronologies

Time granularities, like year, month, and day, play an important role in temporal modeling. We are going to show that time granularities can be modeled in an elegant way by a restricted class of TARs, called *chronologies*.

In nearly all application domains, there is a smallest time unit beyond which measuring time is impossible or meaningless. For example, railway timetables show departure and arrival times of trains with a precision of minutes but not seconds. Intuitively, we think of our time line $\mathbb{N}$ as representing the smallest time unit of the application in hand. In most examples throughout this paper, it is appropriate to think of natural numbers as days: one represents January 1, 1950, two represents January 2, 1950,

[2] Let X, Y be sets with $X \subseteq Y$. We say that X is a computable subset of Y iff there is an algorithm which takes an arbitrary element $y \in Y$ and determines whether or not y is a member of X.

and so on. Then the notion of "month" can be captured by the smallest TAR containing the pair $(i,j) \in \mathit{Future}$ whenever i and j represent days of the same month. That is,

$$\begin{aligned}\texttt{Month} = \{\ & \overbrace{(1,1),\ldots,(1,31),(2,2),\ldots,(2,31),\ldots,(30,30),(30,31),(31,31)}^{January,1950},\\ & \overbrace{(32,32),\ldots,(32,59),(33,33),\ldots,(33,59),\ldots,(58,58),(58,59),(59,59)}^{February,1950},\\ & \ldots,\overbrace{(366,366),\ldots,(396,396)}^{January,1951},\ldots\}\end{aligned}$$

Month satisfies some properties that are typical of granularities in general. First, its symmetric closure is an equivalence relation—every equivalence class represents a single month. And second, months are not "interleaved." A TAR satisfying these properties will be called a *chronology*.

Definition 3 We define a relation $\prec$ on $\wp(\mathbb{N})$. Let $P, Q \subseteq \mathbb{N}$.
P is said to be *before* Q, denoted $P \prec Q$, iff for every $i \in P$, for every $j \in Q$, we have $i < j$. The relation $\prec$ on $\wp(\mathbb{N})$ gives rise to a relation $\preceq$ as follows: $P \preceq Q$ iff $P \prec Q$ or $P = Q$.

Let α be a TAR. We write $dom(\alpha)$ for the smallest set of natural numbers containing i and j whenever $(i,j) \in \alpha$. We write $\alpha^{\mathbf{sym}}$ for the symmetric closure of α.

α is called a *chronology* iff

1. $\alpha^{\mathbf{sym}}$ is an equivalence relation on $dom(\alpha)$, and[3]
2. the set of equivalence classes of $\alpha^{\mathbf{sym}}$, ordered by $\preceq$, is a totally ordered set.

α is called *non-chronological* if it is not a chronology. □

For example, **Month** is a chronology. The equivalence classes of $\texttt{Month}^{\mathbf{sym}}$, ordered by $\preceq$, are $\{1,2,\ldots,31\} \prec \{32,33,\ldots,59\} \prec \ldots$ The notion of chronology captures its intended meaning, namely a partition of the time line in successive time granules. Later on in Section 7.1, we show that chronologies correspond to the notion of *temporal type*, which is used by Wang et al. [29] to model the granularity of time.

3 Trend Dependency (TD)

Having defined the notions of DAS and TAR, we are now ready to introduce the concept of TD. In this section, we first give an impression of the expressiveness of TDs, and then we give a formal definition.

3.1 Motivating Examples

In Section 1, we already explained the meaning of a TD. We now give some additional examples. The constraint *"For an employee, an increase of rank cannot imply a decrease of salary,"* is expressed by the TD

$$(\texttt{SS\#},=)(\texttt{Rank},<) \rightarrow_{Next} (\texttt{Sal},\leq).$$

The constraint *"Changing an employee's rank implies changing his/her salary,"* is expressed by

$$(\texttt{SS\#},=)(\texttt{Rank},\neq) \rightarrow_{Next} (\texttt{Sal},\neq).$$

[3] That is,

- $\forall i \in dom(\alpha) : \alpha^{\mathbf{sym}}(i,i)$ *(Reflexivity)*
- $\forall i,j,k \in dom(\alpha) : (\alpha^{\mathbf{sym}}(i,j) \wedge \alpha^{\mathbf{sym}}(j,k)) \Rightarrow \alpha^{\mathbf{sym}}(i,k)$ *(Transitivity)*

Note that $\alpha^{\mathbf{sym}}$ is symmetric by definition.

TDs encompass the temporal FDs (TFDs) proposed by Wang et al. [29]. For example, *"An employee cannot have two distinct salaries within the same month,"* is expressed by

$$(\texttt{SS\#}, =) \rightarrow_{\texttt{Month}} (\texttt{Sal}, =),$$

where `Month` is defined as in Section 2.2. Let `MoreThanFiveYearsLater` be a TAR containing (i, j) whenever j represents a day that is more than five years later than day i. Then *"The salary of an employee should have increased after five years,"* is expressed by

$$(\texttt{SS\#}, =) \rightarrow_{\texttt{MoreThanFiveYearsLater}} (\texttt{Sal}, <).$$

Let the attribute `Sen` denote the seniority of employees. The TD

$$(\texttt{Rank}, <)(\texttt{Sen}, <) \rightarrow_{Current} (\texttt{Sal}, \leq)$$

expresses that *"If employee x has a lower rank and seniority than employee y, then x cannot earn more than y (at any one time)."* TDs encompass classical FDs. For example, the TD

$$(\texttt{SS\#}, =) \rightarrow_{Current} (\texttt{Sal}, =)$$

expresses that no employee can have two distinct salaries at any one time. The TD is satisfied by a temporal relation if and only if the functional dependency $\texttt{SS\#} \rightarrow \texttt{Sal}$ is satisfied by each "timeslice" of the temporal relation.

The following example is taken from the medical scene. Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation over the set of attributes $\{\texttt{Patient}, \texttt{Diastolic}, \texttt{Systolic}\}$, storing blood pressure readings from patients. A tuple $\{\texttt{Patient} : x, \texttt{Diastolic} : y, \texttt{Systolic} : z\}$ of I_i means that at day i the diastolic blood pressure of patient x was y, and the systolic blood pressure was z. Normally, an increasing diastolic blood pressure implies an increasing systolic blood pressure:

$$(\texttt{Patient}, =)(\texttt{Diastolic}, <) \rightarrow_{Next} (\texttt{Systolic}, <).$$

Note that TDs do not allow comparing different attributes with one another. For example, let the inequality $\texttt{Diastolic} < \texttt{Systolic}$ express the fact that the diastolic blood pressure of a person is always less than his/her systolic blood pressure. This inequality cannot be expressed by a TD. We found that the combination of such inequalities with TDs raises interesting but non-trivial issues. For example, the (non-realistic) TDs $(\texttt{Patient}, =) \rightarrow_{Next} (\texttt{Diastolic}, <)$ and $(\texttt{Patient}, =) \rightarrow_{Next} (\texttt{Systolic}, >)$ taken together express that the diastolic and systolic blood pressure of a patient converge. Then in order to satisfy $\texttt{Diastolic} < \texttt{Systolic}$, patients must disappear from the database at some point in time.

3.2 Syntax and Semantics

We now define the syntax and semantics of TDs. Logical implication captures its classical meaning.

Definition 4 The cardinality of a set S is denoted $|S|$. Let $U \subseteq \mathbf{att}$. A *relation* over U is a finite set of tuples over U. A *temporal relation* over U is an infinite sequence $I = \langle I_1, I_2, \ldots \rangle$ of relations over U satisfying the following condition: there is some $n \in \mathbb{N}$ such that for every $i > n$, $I_i = \{\}$. Each I_i is called a *timeslice* of I. The *cardinality* of a temporal relation I, denoted $|I|$, is equal to $|I_1| + |I_2| + \ldots$ It follows that $|I| \in \mathbb{N}$. The temporal relation I is called *empty* iff $|I| = 0$.

A *trend dependency* (TD) over U (or simply TD, if U is understood) is a statement $\Phi \rightarrow_\alpha \Psi$ where α is a TAR, and Φ, Ψ are DASs with $atts(\Phi), atts(\Psi) \subseteq U$.

Let σ be $\Phi \rightarrow_\alpha \Psi$. We call Φ the *left-hand* DAS, and Ψ the *right-hand* DAS of σ; we say that the TD in hand *involves* the TAR α. We write $tar(\sigma)$ to denote the TAR involved in σ—i.e., $tar(\sigma) = \alpha$.

Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation and let $\Phi \rightarrow_\alpha \Psi$ be a TD (all over U). The TD $\Phi \rightarrow_\alpha \Psi$ is *satisfied* by I iff for every $(i,j) \in \alpha$, for every $s \in I_i$, for every $t \in I_j$, if $\Phi^*(s,t)$ then $\Psi^*(s,t)$.

A TD σ over U is *trivial* iff it is satisfied by each temporal relation over U.

Let Σ be a set of TDs, and let σ be a TD (all over U). We say that the temporal relation $I = \langle I_1, I_2, \ldots \rangle$ over U satisfies Σ iff it satisfies each TD of Σ. We say that Σ logically implies σ, denoted $\Sigma \models \sigma$, iff every temporal relation satisfying Σ also satisfies σ. □

Defining a temporal relation as a time series of snapshot relations is not uncommon in theoretical research. See for example [8, 28]. Also the work on temporal dependency theory of Jensen et al. [20] departs from the idea that temporal relations can be "timesliced." Of course, more advanced structures for storing time-related data have been proposed in the literature. See for example [27, 10]. However, such representation issues are somehow peripheral to this study. Intuitively, one may think of our temporal relations as the result of *timeslicing* an enhanced representation.

4 Logical Implication and Satisfiability Problems

The logical implication and the satisfiability problems for TDs, as for any dependency, are important in practical applications. The problems are the following:

Logical implication problem Given a set Σ of TDs and a TD σ, determine whether $\Sigma \models \sigma$.

Satisfiability problem Given a set Σ of TDs, determine whether Σ can be satisfied in a "non-trivial" way.

The above characterization of the satisfiability problem is still imprecise; a precise formulation will be given later on. For now, note that every set Σ of TDs is trivially satisfied by the empty temporal relation. Logical implication is illustrated by the following example.

Example 2 Let Σ be a set of TDs containing

$$(\texttt{Patient}, =)(\texttt{Diastolic}, <) \rightarrow_{Next} (\texttt{Systolic}, <) \text{ (call it } \sigma_1).$$

We show that Σ logically implies

$$(\texttt{Patient}, =)(\texttt{Systolic}, =) \rightarrow_{Next} (\texttt{Diastolic}, \geq) \text{ (call it } \sigma_2).$$

Suppose $\Sigma \not\models \sigma_2$. Then there is a temporal relation $I = \langle I_1, I_2, \ldots \rangle$ satisfying Σ such that for some $(i,j) \in Next$, for some $s \in I_i$, for some $t \in I_j$, $s(\texttt{Patient}) = t(\texttt{Patient})$ and $s(\texttt{Systolic}) = t(\texttt{Systolic})$ and $s(\texttt{Diastolic}) < t(\texttt{Diastolic})$. Then by σ_1, we have $s(\texttt{Systolic}) < t(\texttt{Systolic})$. Consequently, $s(\texttt{Systolic}) \perp t(\texttt{Systolic})$, a contradiction. We conclude by contradiction that $\Sigma \models \sigma_2$. □

In this section we first define special temporal relations, called *witness temporal relations*, containing at most two tuples. We then define the satisfiability problem, and show it is the dual of the logical implication problem. Finally, we investigate the complexity of both problems.

4.1 Witness Temporal Relations

Witness temporal relations are temporal relations whose cardinality is either 1 or 2. Lemma 2 shows that whenever $\Sigma \not\models \sigma$, then there exists a witness temporal relation satisfying Σ and falsifying σ. Lemma 3 implies that the number of constants appearing in such a witness temporal relation can be limited without loss of generality. This is because in our theoretical framework, any two constants c and d can be related in only three ways: $c < d$, $c = d$, or $c > d$.

Definition 5 Let $(i,j) \in \mathit{Future}$. Let U be a set of attributes. Let s and t be tuples over U. We write $[i \triangleright s, j \triangleright t]$ for the smallest temporal relation $I = \langle I_1, I_2, \ldots \rangle$ over U satisfying: I_i contains s and I_j contains t.[4] Any temporal relation that can be written in this way, is called a *witness temporal relation.* □

Let $I = [i \triangleright s, j \triangleright t]$ be a witness temporal relation. Clearly, if $i \neq j$ or $s \neq t$ then $|I| = 2$; otherwise $|I| = 1$.

Lemma 2 *Let U be a set of attributes. Let Σ be a set of TDs, and let σ be a single TD (all over U). If $\Sigma \not\models \sigma$, then there exists a witness temporal relation satisfying Σ and falsifying σ.*

PROOF. Let $\sigma = \Phi \rightarrow_\alpha \Psi$ and $\Sigma \not\models \sigma$. Hence, there is a temporal relation (call it $I = \langle I_1, I_2, \ldots \rangle$) that satisfies Σ and falsifies σ. That is, for some $(i,j) \in \alpha$, for some $s \in I_i$, for some $t \in I_j$, we have $\Phi^*(s,t)$ and not $\Psi^*(s,t)$. Let I' be $[i \triangleright s, j \triangleright t]$. I' is the desired witness temporal relation. Obviously, I' falsifies σ. We still have to show that I' satisfies Σ. Suppose I' falsifies some TD (call it σ') of Σ. Since I_k contains I'_k for every $k \in \mathbb{N}$, it follows that I must falsify σ', a contradiction. We conclude by contradiction that I' satisfies Σ. This concludes the proof. □

Lemma 3 *Let c, d be constants (i.e., $c, d \in$ **dom**) with $c < d$. Let $(i,j) \in \mathit{Future}$. Let U be a set of attributes. Let s', t' be tuples over U. There exist tuples s, t over U such that*

1. *For every TD σ over U, $[i \triangleright s, j \triangleright t]$ satisfies σ iff $[i \triangleright s', j \triangleright t']$ satisfies σ (i.e., $[i \triangleright s, j \triangleright t]$ and $[i \triangleright s', j \triangleright t']$ satisfy exactly the same TDs), and*
2. *for each $A \in U$, either (a) $s(A) = t(A) = c$, or (b) $s(A) = c$ and $t(A) = d$, or (c) $s(A) = d$ and $t(A) = c$.*

PROOF. Let s, t be tuples over U such that

(a) $s(A) = t(A) = c$ if $s'(A) = t'(A)$,

(b) $s(A) = c$ and $t(A) = d$ if $s'(A) < t'(A)$, and

(c) $s(A) = d$ and $t(A) = c$ if $s'(A) > t'(A)$.

Obviously, for any TD σ, if $[i \triangleright s', j \triangleright t']$ satisfies σ then so does $[i \triangleright s, j \triangleright t]$, and vice versa. This concludes the proof. □

4.2 The Satisfiability Problem

We first give a precise characterization of the satisfiability problem and then we prove that it is the dual of the logical implication problem.

Definition 6 Let U be a set of attributes. Let Σ be a set of TDs over U. Let α be a TAR. We say that Σ is *α-satisfiable* iff for some pair $(i,j) \in \alpha$, for some tuples s, t over U, the witness temporal relation $[i \triangleright s, j \triangleright t]$ satisfies Σ. Otherwise Σ is called *α-unsatisfiable.*

A TDSAT problem is a triple (U, Σ, α) where U is a set of attributes, Σ is a set of TDs over U, and α is a TAR. The answer to the TDSAT problem (U, Σ, α) is "yes" if Σ is α-satisfiable, and "no" otherwise. □

Note that so far we have not specified how infinite TARs are to be represented in a finite way. A finite representation is clearly needed if TARs are part of the input of an effective procedure solving TDSAT. For now we just assume the existence of such a representation. The next subsection will concern the relationship between the representation of TARs and the complexity of TDSAT. We now show the duality between the logical implication problem and TDSAT.

[4] A temporal relation $I = \langle I_1, I_2, \ldots \rangle$ is smaller than a temporal relation $J = \langle J_1, J_2, \ldots \rangle$ if $|I| < |J|$.

Example 3 Suppose we want to decide whether a set Σ of TDs logically implies the TD $(\texttt{SS\#}, =) \rightarrow_{Next} (\texttt{Sal}, \leq)$ (call it σ_1). Let σ_2 be $\emptyset \rightarrow_{Next} (\texttt{SS\#}, =)(\texttt{Sal}, >)$. Assume that $\Sigma \cup \{\sigma_2\}$ is *Next*-satisfiable. That is, for some $(i, j) \in Next$, for some tuples s and t, the witness temporal relation $I = [i \triangleright s, j \triangleright t]$ satisfies Σ as well as σ_2. Since $\emptyset^*(s, t)$ is trivial, we have $s(\texttt{SS\#}) = t(\texttt{SS\#})$ and $s(\texttt{Sal}) > t(\texttt{Sal})$. Then I is obviously a counterexample for $\Sigma \models \sigma_1$. So if $\Sigma \cup \{\sigma_2\}$ is *Next*-satisfiable, then Σ does not logically imply σ_1. Conversely, it can be readily seen that if Σ does not logically imply σ_1, then $\Sigma \cup \{\sigma_2\}$ is *Next*-satisfiable. □

Theorem 1 *Let U be a set of attributes. Let Σ be a set of TDs, and let $\Phi \rightarrow_\alpha \Psi$ be a TD (all over U). $\Sigma \models \Phi \rightarrow_\alpha \Psi$ iff for every pair $(A, \theta) \in \Psi$, the answer to the TDSAT problem (U, Σ', α) is "no" where $\Sigma' = \Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \overline{\theta})\}$.*

Proof. Note that the proof is trivial if $\Psi = \emptyset$.
$\Rightarrow$. Assume for some $(A, \theta) \in \Psi$, $\Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \overline{\theta})\}$ is α-satisfiable. Then for some pair $(i, j) \in \alpha$, for some tuples s, t over U, the witness temporal relation $[i \triangleright s, j \triangleright t]$ satisfies $\Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \overline{\theta})\}$. Since $\emptyset^*(s, t)$ is trivial, we have $\Phi^*(s, t)$ and not $\Psi^*(s, t)$. So $[i \triangleright s, j \triangleright t]$ is a counterexample for $\Sigma \models \Phi \rightarrow_\alpha \Psi$.
$\Leftarrow$. Conversely, assume $\Sigma \not\models \Phi \rightarrow_\alpha \Psi$. By Lemma 2, there exists a witness temporal relation, say $[i \triangleright s, j \triangleright t]$, satisfying Σ and falsifying $\Phi \rightarrow_\alpha \Psi$. Clearly, $(i, j) \in \alpha$, and $\Phi^*(s, t)$, and not $\Psi^*(s, t)$. Hence, for some $(A, \theta) \in \Psi$, $[i \triangleright s, j \triangleright t]$ satisfies $\Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \overline{\theta})\}$. This concludes the proof. □

Conversely, every satisfiability problem is the dual of a logical implication problem, as shown next.

Corollary 1 *Let U be a set of attributes. Let Σ be a set of TDs over U. Let $A \in U$. The answer to the TDSAT problem (U, Σ, α) is "yes" iff $\Sigma \not\models \emptyset \rightarrow_\alpha (A, \bot)$.*

Proof. By Theorem 1, $\Sigma \not\models \emptyset \rightarrow_\alpha (A, \bot)$ iff $\Sigma \cup \{\emptyset \rightarrow_\alpha (A, \top)\}$ is α-satisfiable. Since $\emptyset \rightarrow_\alpha (A, \top)$ is a trivial TD, we have $\Sigma \not\models \emptyset \rightarrow_\alpha (A, \bot)$ iff Σ is α-satisfiable. This concludes the proof. □

4.3 Complexity

The following theorem considers the complexity of the logical implication problem for TDs.

Theorem 2 *The logical implication problem for TDs is* **coNP***-hard.*

Proof. TDs encompass typed clausal constraint-generating 2-dependencies, proposed by Baudinet et al. [5]. The logical implication problem is **coNP**-complete for this type of dependencies [5, theorem 10]. This concludes the proof. □

As an immediate corollary of Theorem 1 and Theorem 2, TDSAT is **NP**-hard. We now explore an upper bound for the complexity of a given TDSAT problem (U, Σ, α). As one may expect, this complexity depends on the formalism used to represent TARs. A TAR was defined as a possibly infinite subset of *Future* (Definition 2). Of course, a real system can only deal with TARs that have a finite representation. For example, the infinite TAR *Next* can be represented in a finite way by the equality $y = x + 1$ where x and y are interpreted over $\mathbb{N}$. We are now going to study the relationship between the complexity of TDSAT and the formalism chosen to represent TARs.

Consider a TD $(\texttt{SS\#}, =) \rightarrow_{Current} (\texttt{Sal}, <)$, stating that an employee earns less than his/her salary. Obviously, the only way to satisfy this TD is by the empty temporal relation. This observation is captured and generalized by the following definition.

Definition 7 Let U be a set of attributes. Let Σ be a set of TDs over U. A natural number $i \in \mathbb{N}$ is called *improper* w.r.t. Σ iff Σ contains some TD $\Phi \rightarrow_\alpha \Psi$ where Φ is reflexive, Ψ is not reflexive, and $(i, i) \in \alpha$; otherwise i is said to be *proper* w.r.t. Σ. We write $prop(\Sigma)$ for the smallest set containing $(i, j) \in Future$ whenever i and j are proper w.r.t. Σ. Σ is called *consistent* iff $prop(\Sigma) = Future$; otherwise it is called *inconsistent*. □

It follows that the singleton set $\{(\mathtt{SS\#}, =) \rightarrow_{Current} (\mathtt{Sal}, <)\}$ is inconsistent (every natural number is inconsistent w.r.t. this set). A similar notion of impropriety is used in [14]. The following lemma states that an inconsistent set of TDs requires certain timeslices to be empty.

Lemma 4 *Let U be a set of attributes. Let Σ be a set of TDs over U. A natural number i is improper w.r.t. Σ iff for every temporal relation $I = \langle I_1, I_2, \ldots \rangle$ over U that satisfies Σ, we have $I_i = \{\}$.*

PROOF. $\Rightarrow$. Let $i \in \mathbb{N}$. Assume i is improper w.r.t. Σ. Hence, Σ contains some TD $\Phi \rightarrow_\alpha \Psi$ where Φ is reflexive, Ψ is not reflexive, and $(i,i) \in \alpha$. Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation satisfying Σ. It suffices to show that $I_i = \{\}$. Suppose the opposite—i.e., I_i contains a tuple (call it t) over U. By Lemma 1, $\Phi^*(t,t)$ and not $\Psi^*(t,t)$. Since $(i,i) \in \alpha$, I falsifies $\Phi \rightarrow_\alpha \Psi$, a contradiction. We conclude by contradiction that $I_i = \{\}$. $\Leftarrow$. Suppose i is proper. Let t be a tuple over U. Let $I = [i \triangleright t, i \triangleright t]$. It is easy to see that I satisfies Σ and $I_i = \{t\} \neq \{\}$. This concludes the proof. □

In a first naive attempt to solve a TDSAT problem (U, Σ, α) one could try all tuples s, t over U, and all pairs $(i,j) \in \alpha$, and verify whether $[i \triangleright s, j \triangleright t]$ satisfies Σ. By Lemma 3, it suffices to try at most $3^{|U|}$ tuple pairs (s,t). By Lemma 4, it suffices to try only pairs (i,j) of α that also belong to $prop(\Sigma)$—for if i or j is improper w.r.t. Σ, then $[i \triangleright s, j \triangleright t]$ falsifies Σ. That is, the TDSAT problem (U, Σ, α) can be reduced to the TDSAT problem $(U, \Sigma, \alpha \cap prop(\Sigma))$. Nevertheless, trying all pairs (i,j) of $\alpha \cap prop(\Sigma)$ still poses severe problems, as $\alpha \cap prop(\Sigma)$ can be infinite, or there may be no effective method for obtaining all members of it.

Fortunately, there is no need to try all pairs (i,j) of $\alpha \cap prop(\Sigma)$, as we are going to show. This is because we can partition $prop(\Sigma)$ into a finite number of *homogeneous* subsets, in the sense that whenever (i,j) and (k,l) belong to the same subset, then $[i \triangleright s, j \triangleright t]$ and $[k \triangleright s, l \triangleright t]$ *either* both satisfy Σ, *or* both falsify Σ. Consequently, in trying pairs (i,j) of $\alpha \cap prop(\Sigma)$, we never need to try two pairs belonging to the same homogeneous subset.

Definition 8 Let Σ be a set of TDs. A TAR α is said to be *homogeneous* w.r.t. Σ iff for all tuples s, t over U, *either* for all $(i,j) \in \alpha$, $[i \triangleright s, j \triangleright t]$ satisfies Σ, *or* for all $(i,j) \in \alpha$, $[i \triangleright s, j \triangleright t]$ falsifies Σ.

Let $n \in \mathbb{N}$. We write $\mathcal{F}_n$ for the smallest set containing every total function from $[0..n]$ to $\{\mathtt{neg}, \mathtt{pos}\}$. Obviously, $|\mathcal{F}_n| = 2^{n+1}$.

Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a *list* of TDs.[5] Let $f \in \mathcal{F}_n$. The TAR *induced by* f and Σ, denoted $[\![f]\!]_\Sigma$, is defined as follows:

$$[\![f]\!]_\Sigma = prop(\Sigma) \cap \beta_0 \cap \beta_1 \cap \ldots \cap \beta_n$$

where

- $\beta_0 = Current$ if $f(0) = \mathtt{pos}$ and $\beta_0 = \neg Current$ if $f(0) = \mathtt{neg}$, and [6]
- for every $i \in [1..n]$, $\beta_i = tar(\sigma_i)$ if $f(i) = \mathtt{pos}$ and $\beta_i = \neg tar(\sigma_i)$ if $f(i) = \mathtt{neg}$.

□

The concepts of homogeneity and induced TAR are illustrated by Example 4. Lemma 5 states that $\mathcal{F}_n$ induces a partitioning of $prop(\Sigma)$ with $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$. More precisely, for every $(i,j) \in prop(\Sigma)$, there is some $f \in \mathcal{F}_n$ such that $(i,j) \in [\![f]\!]_\Sigma$. As a corollary, the answer to the TDSAT problem $(U, \Sigma, \alpha \cap prop(\Sigma))$ is "yes" if and only if the answer to the TDSAT problem $(U, \Sigma, \alpha \cap [\![f]\!]_\Sigma)$ is "yes" for some $f \in \mathcal{F}_n$. So $\mathcal{F}_n$ induces a decomposition of the TDSAT problem (U, Σ, α) into a number of new TDSAT problems. Interestingly, Lemma 6 states that each TAR $[\![f]\!]_\Sigma$ is homogeneous w.r.t. Σ.

[5] By saying that Σ is a list, we mean that the left-to-right numbering of the TDs in Σ is relevant.

[6] Recall from Definition 2 that $\neg\alpha = Future \setminus \alpha$.

Example 4 Let $\Sigma = \{\sigma_1, \sigma_2\}$ where $\sigma_1 = (\texttt{SS\#}, =)(\texttt{Rank}, <) \rightarrow_{Next} (\texttt{Sal}, <)$ and $\sigma_2 = (\texttt{Rank}, <) \rightarrow_{Current} (\texttt{Sal}, <)$. That is, $tar(\sigma_1) = Next$ and $tar(\sigma_2) = Current$. $\mathcal{F}_2$ contains eight functions; one element of $\mathcal{F}_2$ is $\{(0, \texttt{neg}), (1, \texttt{pos}), (2, \texttt{neg})\}$ (call it f). The TAR induced by f and Σ is given by

$$[\![f]\!]_\Sigma = prop(\Sigma) \cap \neg Current \cap Next \cap \neg Current,$$

which happens to be equal to *Next*. Let s, t be two employee tuples. If $s(\texttt{SS\#}) = t(\texttt{SS\#})$ and $s(\texttt{Rank}) < t(\texttt{Rank})$ and $s(\texttt{Sal}) \geq t(\texttt{Sal})$ then $[i \triangleright s, j \triangleright t]$ falsifies Σ for *every* pair (i, j) of *Next*. On the other hand, if $s(\texttt{SS\#}) \neq t(\texttt{SS\#})$ or $s(\texttt{Rank}) \geq t(\texttt{Rank})$ or $s(\texttt{Sal}) < t(\texttt{Sal})$ then $[i \triangleright s, j \triangleright t]$ satisfies Σ for *every* pair (i, j) of *Next*. From this, it is correct to conclude that *Next* is homogeneous w.r.t. Σ. □

Lemma 5 *Let U be a set of attributes. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a list of TDs over U. Then $prop(\Sigma) = \bigcup\{[\![f]\!]_\Sigma \mid f \in \mathcal{F}_n\}$.*

Proof. Straightforward. □

Lemma 6 *Let U be a set of attributes. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a list of TDs over U. Let $f \in \mathcal{F}_n$. Then $[\![f]\!]_\Sigma$ is homogeneous w.r.t. Σ. Moreover, for all tuples s, t over U, it can be decided in constant time in the size of $[\![f]\!]_\Sigma$ whether for all $(k, l) \in [\![f]\!]_\Sigma$, $[k \triangleright s, l \triangleright t]$ satisfies Σ.*

Proof. The proof is trivial if $[\![f]\!]_\Sigma = \{\}$. Next assume $[\![f]\!]_\Sigma \neq \{\}$. Let s, t be two tuples over U. Let $(k, l) \in [\![f]\!]_\Sigma$. Let $i \in [1..n]$. Let σ_i be $\Phi \rightarrow_\alpha \Psi$. Then $[k \triangleright s, l \triangleright t]$ satisfies σ_i if and only if one of the following conditions is satisfied:

1. $f(i) = \texttt{neg}$ (i.e., $(k, l) \notin \alpha$).
2. $f(i) = \texttt{pos}$ and $f(0) = \texttt{neg}$ (i.e., $(k, l) \in \alpha$ and $k \neq l$), and if $\Phi^*(s, t)$ then $\Psi^*(s, t)$.
3. $f(i) = \texttt{pos}$ and $f(0) = \texttt{pos}$ (i.e., $(k, l) \in \alpha$ and $k = l$), and *(a)* if $\Phi^*(s, t)$ then $\Psi^*(s, t)$, and *(b)* if $\Phi^*(t, s)$ then $\Psi^*(t, s)$.

One may conjecture that σ_i could still be falsified by $[k \triangleright s, l \triangleright t]$ if $(k, k) \in \alpha$ and $\Phi^*(s, s)$ but not $\Psi^*(s, s)$. However, in that case k would be improper w.r.t. Σ, hence $(k, l) \notin [\![f]\!]_\Sigma$, a contradiction. Clearly, if one of the above three conditions is satisfied for some $(k, l) \in [\![f]\!]_\Sigma$, it is satisfied for all $(k, l) \in [\![f]\!]_\Sigma$. Since σ_i is an arbitrary TD of Σ, it is correct to conclude that *either* for all $(k, l) \in [\![f]\!]_\Sigma$, $[k \triangleright s, l \triangleright t]$ satisfies Σ, *or* for all $(k, l) \in [\![f]\!]_\Sigma$, $[k \triangleright s, l \triangleright t]$ falsifies Σ. Since s and t are arbitrary, it is correct to conclude that $[\![f]\!]_\Sigma$ is homogeneous w.r.t. Σ. It can be readily seen that testing conditions (1), (2), and (3) is independent of the size of $[\![f]\!]_\Sigma$. This concludes the proof. □

Suppose we are given the TDSAT problem (U, Σ, α), where $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$. By Lemmas 4 and 5, the answer to (U, Σ, α) is "yes" if and only if the answer to the TDSAT problem $(U, \Sigma, \alpha \cap [\![f]\!]_\Sigma)$ is "yes" for some $f \in \mathcal{F}_n$. By Lemmas 6 and 3, a *non-deterministic* polynomial algorithm can guess $f \in \mathcal{F}_n$ as well as tuples s and t, and determine in polynomial time whether $[k \triangleright s, l \triangleright t]$ satisfies Σ for *all* $(k, l) \in [\![f]\!]_\Sigma$—and hence for all $(k, l) \in [\![f]\!]_\Sigma \cap \alpha$. But remark: Even if this algorithm successfully ends by finding some s, t, and f such that $[k \triangleright s, l \triangleright t]$ satisfies Σ for all $(k, l) \in [\![f]\!]_\Sigma \cap \alpha$, then this does *not* imply that the answer to (U, Σ, α) is "yes." This is because we still have to determine whether $[\![f]\!]_\Sigma \cap \alpha$ is non-empty. This is exactly the point where the representation of TARs comes into play. Since we did not specify the representation of TARs, there is little specific we can say about the complexity of deciding non-emptiness of $[\![f]\!]_\Sigma \cap \alpha$. The following theorem may provide an upper bound of the complexity in many practical applications, however.

Theorem 3 *Let TARINTERSECT be the following problem: Given a set $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ of TDs, a TAR α, and some $f \in \mathcal{F}_n$, determine whether $[\![f]\!]_\Sigma \cap \alpha$ is non-empty. If TARINTERSECT is in* **P**, *then TDSAT is in* **NP**.

	TD	Justification
1.	$(\texttt{Patient},=)(\texttt{Diastolic},<) \rightarrow_{Next} (\texttt{Systolic},<)$	given.
2.	$(\texttt{Patient},=)(\texttt{Diastolic},<)(\texttt{Systolic},=) \rightarrow_{Next} (\texttt{Systolic},\perp)$	from 1 by TD1.
3.	$(\texttt{Systolic},\perp) \rightarrow_{Future} (\texttt{Diastolic},\perp)$	by TD7.
4.	$(\texttt{Patient},=)(\texttt{Diastolic},<)(\texttt{Systolic},=) \rightarrow_{Next} (\texttt{Diastolic},\perp)$	from 2 and 3 by TD2.
5.	$(\texttt{Patient},=)(\texttt{Systolic},=) \rightarrow_{Next} (\texttt{Diastolic},\geq)$	from 4 by TD5.

Figure 2: Example derivation.

PROOF. Let TARINTERSECT be in **P**. By Lemmas 4 and 5, the answer to (U, Σ, α) is "yes" if and only if the answer to the TDSAT problem $(U, \Sigma, \alpha \cap [\![f]\!]_\Sigma)$ is "yes" for some $f \in \mathcal{F}_n$. By Lemmas 6 and 3, a *non-deterministic* polynomial algorithm can guess $f \in \mathcal{F}_n$ as well as tuples s and t, and determine in polynomial time whether $[\![f]\!]_\Sigma \cap \alpha$ is non-empty (an instance of TARINTERSECT), and, if so, whether $[k \triangleright s, l \triangleright t]$ satisfies Σ for *all* $(k, l) \in [\![f]\!]_\Sigma$. □

It is likely that in many practical applications, TARINTERSECT will be in **P**. This is the case in situations where there is a finite number of fixed TARs (time granularities, for example) and solutions to TARINTERSECT can be tabulated (for example, $\texttt{Month} \cap \neg\texttt{Year} = \{\}$ since two times of the same month must belong to the same year).

5 Axiomatization

In this section, we give a sound and complete axiomatization for logical implication of TDs. The axiomatization captures concisely the essential properties of TDs.

Definition 9 Let U be a set of attributes.
Let Φ, Ψ, Υ be DASs with $atts(\Phi), atts(\Psi), atts(\Upsilon) \subseteq U$. Let $A \in U$. Let α, β be TARs.

TD1 If $\Phi \rightarrow_\alpha \Psi$ then $\Phi \wedge \Upsilon \rightarrow_\alpha \Psi \wedge \Upsilon$. *(Augmentation)*

TD2 If $\Phi \rightarrow_\alpha \Psi$ and $\Psi \rightarrow_\beta \Upsilon$ then $\Phi \rightarrow_{\alpha\cap\beta} \Upsilon$. *(Transitivity)*

TD3 If $\Phi \rightarrow_\alpha \Psi$ and $\Phi \rightarrow_\beta \Psi$ then $\Phi \rightarrow_{\alpha\cup\beta} \Psi$. *(Upward heredity)*

TD4 $\Phi \rightarrow_{\{\}} \Psi$. *(Emptiness)*

TD5 If $\Phi \wedge (A, \theta) \rightarrow_\alpha (A, \eta)$ then $\Phi \rightarrow_\alpha (A, \overline{\theta} \cup \eta)$. *(Simplification)*

TD6 If $\Phi \rightarrow_\alpha \Psi$ then $\widehat{\Phi} \rightarrow_{\alpha\cap Current} \widehat{\Psi}$. *(Reciprocity)*

TD7 $(A, \perp) \rightarrow_{Future} \Phi$. *(False premise)*

TD8 If $\alpha \subseteq \beta$ and $\Phi \rightarrow_\beta \Psi$ then $\Phi \rightarrow_\alpha \Psi$. *(Downward heredity)*

Let Σ be a set of TDs over U. Let σ be a TD over U. We write $\Sigma \vdash \sigma$ to denote that σ is provable from Σ using the above axiomatization. Here the notion of proof captures its classical meaning [1, page 167]). □

Example 5 Figure 2 illustrates how the TD

$$(\texttt{Patient},=)(\texttt{Systolic},=) \rightarrow_{Next} (\texttt{Diastolic},\geq)$$

can be proved from

$$\{(\texttt{Patient},=)(\texttt{Diastolic},<) \rightarrow_{Next} (\texttt{Systolic},<)\}.$$

□

I_1 :

SS#	Rank	Sal
A1	1	100
B2	1	120
C3	3	140
D4	2	80
E5	2	120

I_2 :

SS#	Rank	Sal
A1	2	110
B2	2	110
C3	2	130
D4	3	90
E5	3	120

Figure 3: Example database.

The following theorem states the soundness and the completeness of the axiomatization for consistent sets of TDs. The proof is lengthy and constitutes the content of [32].

Theorem 4 *Let U be a set of attributes. Let Σ be a consistent set of TDs over U, and let σ be a TD (all over U). $\Sigma \models \sigma$ iff $\Sigma \vdash \sigma$.*

6 TD Mining

As TDs allow expressing significant knowledge about the data stored in a database, discovering them from existing databases is interesting and important. Knowledge discovery in databases, also called data mining, is currently recognized as a promising research area with a high application potential. The data mining problem we are going to study can be loosely described as follows: Given a temporal relation, find the TDs that are satisfied to "a high degree," and that conform to a given template, which fixes the attributes to be used and the TAR involved.

The outline of this section is as follows. The next subsection contains an introductory example. In Section 6.2, we give a formal definition of the data mining problem TDMINE. Its complexity is studied in Section 6.3, and algorithmic aspects are discussed in Section 6.4.

6.1 Introductory Example

The notion of satisfaction we have used so far is a "black-and-white" concept: Given a temporal relation $I = \langle I_1, I_2, \ldots \rangle$, a TD σ is either satisfied or falsified by I; there is no third possibility. Such a black-and-white approach to satisfaction is not very appropriate for data mining purposes. In data mining, one is typically not only interested in the rules that are fully satisfied, but also in those that are "highly" satisfied. What we need is a notion of gradual satisfaction. For the purpose of TD mining, we characterize TD satisfaction by the notions of *support* and *confidence*, which are common in the work on association rule mining. See for example [3, 34].

Consider the temporal relation $I = \langle I_1, I_2, \ldots \rangle$ shown in Figure 3; it is understood that $I_3, I_4, \ldots$ are all empty. Assume no temporal constraints have been specified about the evolution of salaries. Consider the TD

$$\sigma = (\texttt{SS\#}, =)(\texttt{Rank}, <) \rightarrow_{NextOne} (\texttt{Sal}, \leq)$$

expressing that if the rank of an employee increases between time 1 and 2, then his/her salary does not decrease. Employees A1, D4, and E5 support the trend, as their rank increased and their salary did not decrease. Employee B2 gives evidence against the trend, showing a rank increase together with a salary decrease. Finally, employee C3 provides no argument for or against the trend, as his/her rank did not increase.

We now quantify the above observations. Every tuple pair of $I_1 \times I_2$ satisfying the left-hand DAS of σ, gives evidence for or against the TD in hand. The tuple pairs satisfying both the left-hand and the right-hand DAS are said to support the TD. The confidence c is obtained by the number of tuple pairs supporting the TD divided by the number of tuple pairs satisfying the left-hand DAS; in the example $c = 3/4$. Note that since the confidence is expressed as a proportion, it can be close to one, even though, in absolute terms, there are actually few tuple pairs supporting the TD. Therefore an additional measure is needed to characterize the importance of a given TD. The support s is the number of tuple pairs

supporting the TD divided by the cardinality of $I_1 \times I_2$; in the example $s = 3/25$. So σ is satisfied with support 3/25, and with confidence 3/4.

Next consider the TD

$$\sigma' = (\texttt{SS\#}, =)(\texttt{Rank}, <) \rightarrow_{NextOne} (\texttt{Sal}, >)$$

which expresses the opposite trend that the salary of an employee decreases if his/her rank increases (`Rank` could be a measure of malpractice, rather than performance!). This TD is satisfied with support 1/25, and with confidence 1/4. During TD mining, σ is to be preferred above σ' because σ is satisfied with a higher support and confidence.

6.2 The TD Mining Problem

We now define the notions of *support* and *confidence*. The definition conforms to the intuition given in the previous subsection.

Definition 10 Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation over the set U of attributes. Let $\Phi \rightarrow_\alpha \Psi$ be a TD over U. Let $(i,j) \in \alpha$.
Let $L_{(i,j)} = \{(s,t) \in I_i \times I_j \mid \Phi^*(s,t)\}$.
Let $B_{(i,j)} = \{(s,t) \in I_i \times I_j \mid \Phi^*(s,t) \text{ and } \Psi^*(s,t)\}$.
Let

$$p = \sum_{(i,j)\in\alpha} |I_i \times I_j| \text{ and } l = \sum_{(i,j)\in\alpha} |L_{(i,j)}| \text{ and } b = \sum_{(i,j)\in\alpha} |B_{(i,j)}|.$$

Let

$$s = \begin{cases} b/p & \text{if } p \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } c = \begin{cases} b/l & \text{if } l \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then $\Phi \rightarrow_\alpha \Psi$ is said to be *satisfied* by I with *support* s and *confidence* c, denoted $I \models^s_c \Phi \rightarrow_\alpha \Psi$. We also say that s and c are the support and the confidence of $\Phi \rightarrow_\alpha \Psi$ respectively (where I is implicitly understood). Clearly, $0 \leq s \leq c \leq 1$. □

It can be readily seen that if the confidence of a TD is equal to one, then it is satisfied in the sense of Definition 4. We next define the notions of TD *template* and TD *class*.

Definition 11 A TD *template* over the set U of attributes is a statement $X \rightarrow_\alpha Y$ where $X, Y \subseteq U$ and α is a TAR.

We define $\mathbf{op} = \mathbf{OP} \setminus \{\perp, \top\}$. Let Θ be a non-empty subset of $\mathbf{op}$. The TD *class* determined by the TD template $X \rightarrow_\alpha Y$ and Θ, denoted $[\![X \rightarrow_\alpha Y]\!]^\Theta$, is the smallest set of TDs containing the TD $\Phi \rightarrow_\alpha \Psi$ whenever

- $atts(\Phi) = X$ and for every $A \in X$, $\Phi(A) \in \Theta$, and
- $atts(\Psi) = Y$ and for every $A \in Y$, $\Psi(A) \in \Theta$.

The Greek lowercase letter τ will be used for TD templates. $[\![\tau]\!]$ is a shorthand for $[\![\tau]\!]^{\mathbf{op}}$. □

For example, let A and B be attributes, and let τ denote the TD template $\{A\} \rightarrow_\alpha \{B\}$. Then $[\![\tau]\!]^{\{\leq,\geq\}}$ contains four TDs, among others, $(A, \geq) \rightarrow_\alpha (B, \leq)$. Remark: We are no longer interested in the operators $\top$ and $\perp$ for the following reasons. The TD $(A_1, \top) \ldots (A_m, \top) \rightarrow_\alpha (A_{m+1}, \top) \ldots (A_n, \top)$ is always satisfied with support one and confidence one. Such a TD would not be an interesting outcome of a data mining process. On the other hand, if a TD σ involves $\perp$, then it is satisfied with support zero and confidence zero. Again, such a TD is of no interest to data mining.

The data mining problem TDMINE we are going to explore can now be defined.

Definition 12 We use $(0,1)$ to denote the set of real numbers $r \in \mathbb{R}$ with $0 \leq r \leq 1$.

Let Θ be a non-empty subset of **op**. A TDMINE$_\Theta$ problem is a quintet (U, I, τ, ts, tc) where U is a set of attributes, $I = \langle I_1, I_2, \ldots \rangle$ is a temporal relation over U, $ts, tc \in (0, 1)$, and τ is a TD template satisfying (let $\tau = X \rightarrow_\alpha Y$):

$$\alpha = \{(i_1, j_1), \ldots, (i_m, j_m)\}$$

where $m \geq 0$ and $(i_k, j_k) \in \mathit{Future}$ $(k \in [1..m])$. That is, α is given as a finite set of members of *Future*.

The solution to the TDMINE$_\Theta$ problem (U, I, τ, ts, tc) is the smallest set of TDs over U containing σ iff (let $s, c \in (0, 1)$ such that $I \models^s_c \sigma$):

- $\sigma \in [\![\tau]\!]^\Theta$,
- $s \geq ts$ *(threshold support)*, and
- $c \geq tc$ *(threshold confidence)*.

TDMINE is a shorthand for TDMINE$_{\mathbf{op}}$.

If a TDMINE problem (U, I, τ, ts, tc) is implicitly understood from the context, we use the following syntactic shorthands for characterizing its input size:

- $\mathcal{C}$ denotes the cardinality of I. That is, $\mathcal{C} = |I|$.
- $\mathcal{N}$ denotes a time (in practice, the smallest time) satisfying $I_i = \{\}$ for each $i > \mathcal{N}$. So $I = \langle I_1, I_2, \ldots \rangle$ is fully determined by $\langle I_1, I_2, \ldots, I_\mathcal{N} \rangle$.
- $\mathcal{D}$ (degree) denotes the number of attributes occurring in τ. That is, if $\tau = X \rightarrow_\alpha Y$, then $\mathcal{D} = |XY|$.[7]

□

So the TDMINE$_\Theta$ problem (U, I, τ, ts, tc) is the task of finding all TDs of $[\![\tau]\!]^\Theta$ that are satisfied by I with support $\geq ts$ and confidence $\geq tc$. Instead of defining the solution as the set of *all* TDs whose support and confidence exceed certain threshold values, one could limit the solution to the TDs that optimize either the support or the confidence. This alternative definition would not affect the results presented in this section.

In Section 4.3, we indicated that the formalism used to represent TARs determines the complexity of reasoning about TDs. Definition 12 requires that the TAR α which occurs in a TDMINE$_\Theta$ problem, is given by finite enumeration. This requirement is not a strong one, as we explain. Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation over the set U of attributes, and let $\mathcal{N}$ be a natural number such that $I_i = \{\}$ for each $i > \mathcal{N}$. Let β be a (possibly infinite) TAR represented in one formalism or another. Let $\tau = X \rightarrow_\beta Y$ be a TD template over U. Let $ts, tc \in (0, 1)$. Let S be the smallest set of TDs containing every TD of $[\![\tau]\!]^\Theta$ that is satisfied by I with support $\geq ts$ and confidence $\geq tc$. Assume we are interested in computing S. Note that the task of computing S differs from TDMINE$_\Theta$ because β may not be finite. Nevertheless, this task can be readily reduced to a TDMINE$_\Theta$ problem, as follows. Let $\eta = \{(i, j) \in \mathit{Future} \mid j \leq \mathcal{N}\}$, a finite TAR. Let $\eta \cap \beta = \{(i_1, j_1), \ldots, (i_m, j_m)\}$ $(m \geq 0)$. Note that $\eta \cap \beta$ can be easily computed by an algorithm that generates each member of η in turn and decides whether or not it belongs to β. Then the TD $\Phi \rightarrow_\beta \Psi$ belongs to S if and only if the TD $\Phi \rightarrow_{\eta \cap \beta} \Psi$—with the same left-hand and right-hand DAS—belongs to the solution of the TDMINE$_\Theta$ problem (U, I, τ, ts, tc) with $\tau = X \rightarrow_{\eta \cap \beta} Y$. This is because $I_i \times I_j = \{\}$ if $(i, j) \in \beta \setminus \eta$. So S can readily be derived from the solution of (U, I, τ, ts, tc). To conclude, the practical constraint imposed on the input TAR of a TDMINE problem does not decrease the generality of the problem.

We now define a decision problem, called TDMINE(D), which is intimately related to TDMINE.

[7] Concatenation is used for union. That is, $XY = X \cup Y$.

Definition 13 Let Θ be a non-empty subset of **op**. A TDMINE(D)$_\Theta$ problem is a quintet (U, I, τ, ts, tc) where U, $I = \langle I_1, I_2, \ldots \rangle$, τ, ts, and tc are as in Definition 12.

The solution to (U, I, τ, ts, tc) is "yes" or "no" depending on whether or not there exists some TD $\sigma \in [\![\tau]\!]^\Theta$ such that $I \models_c^s \sigma$ for some $s \geq ts$ and $c \geq tc$. TDMINE(D) is a shorthand for TDMINE(D)$_{\mathbf{op}}$. □

So TDMINE(D) asks whether a specified support and confidence can be attained by some TD of a given TD class. Obviously, TDMINE$_\Theta$ is at least as hard as TDMINE(D)$_\Theta$: If we have a polynomial-time algorithm for TDMINE$_\Theta$, then we certainly do for TDMINE(D)$_\Theta$. However, it turns out that TDMINE(D)$_\Theta$ is **NP**-complete for certain Θ, as we show in the next subsection.

TDMINE(D)$_\Theta$ can be solved in a brute force manner by an exhaustive algorithm that computes the support and the confidence of each TD in $[\![\tau]\!]^\Theta$. The number of TDs in $[\![\tau]\!]^\Theta$ is $\mathcal{O}(|\Theta|^{\mathcal{D}})$. The confidence and the support of a given TD $\Phi \rightarrow_\alpha \Psi$ can be computed in quadratic time in $\mathcal{C}$, as follows. For each $(i,j) \in \alpha$, we compute $L_{(i,j)}$ and $B_{(i,j)}$ as defined in Definition 10 by comparing all tuples of I_i with all tuples of I_j. The confidence and support can be computed from the summation of $L_{(i,j)}$ and $B_{(i,j)}$ over all $(i,j) \in \alpha$. In the worst case, we have to compare every tuple of I with every other tuple of I, or $\mathcal{O}(\mathcal{C}^2)$ comparisons.

6.3 Complexity

In this section we explore the complexity of TDMINE(D)$_\Theta$. This leads to the following interesting and important results:

- TDMINE(D) is **NP**-complete.
- TDMINE(D)$_{\{<,=,>\}}$, on the other hand, is in **P**.

Algorithmic aspects will be discussed in the next section.

Lemma 7 *Let U be a set of attributes. Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation over U. Let $\Phi \rightarrow_\alpha (A, \theta)$ be a TD over U with $\theta \in$ **op**. Let $s, c \in (0,1)$. If $I \models_c^s \Phi \rightarrow_\alpha (A, \theta)$ then for some $\theta' \in \{\leq, \geq, \neq\}$, for some $s' \geq s$ and $c' \geq c$, we have $I \models_{c'}^{s'} \Phi \rightarrow_\alpha (A, \theta')$.*

Proof. Straightforward. □

Theorem 5 TDMINE(D) *is* **NP***-complete.*

Proof. TDMINE(D) can be solved by a *non-deterministic* polynomial algorithm, one that guesses a TD of the specified TD class and computes the support and confidence in polynomial time, and then checks whether these values exceed the specified minimum thresholds; hence TDMINE(D) is in **NP**. We now prove that 3SAT can be reduced to TDMINE(D). Consider the propositional formula:

$$\Delta = \bigwedge_{i=1..m} \chi_{i1} \vee \chi_{i2} \vee \chi_{i3}$$

where each χ_{ij} is either a variable or the negation of one. Let $V = \{x_1, x_2, \ldots, x_v\}$ be the smallest set containing each variable appearing in Δ. $v \geq 3$ is assumed without loss of generality. Let U be a set of attributes. For convenience, we assume $U = V \cup \{r\}$ where $r \notin V$. We describe the reduction R next.

We assume without loss of generality that $0 < 0.5 < 1$ are three constants of **dom**. Let $x \in U$. We write $t_{x=a}$ for the tuple t over U satisfying: $t(x) = a$, and $t(y) = 0.5$ if $y \neq x$. Let I_1 be a singleton containing $t_{r=0.5}$. Let I_{21} be the smallest relation over U containing $t_{x=0}$ and $t_{x=1}$ for every $x \in V$.

For every $i \in [1..m]$ we define three tuples (denoted t_{i0}, t_{i1}, and t_{i2}), with for each $x \in U$, for each $j \in \{0,1,2\}$,

	x_1	x_2	x_3	...	x_v	r	
I_1 :	.5	.5	.5	...	.5	.5	
I_{21} :	0	.5	.5	...	.5	.5	
	.5	0	.5	...	.5	.5	
	.5	.5	0	...	.5	.5	
				...			
	.5	.5	.5	...	0	.5	
	1	.5	.5	...	.5	.5	
	.5	1	.5	...	.5	.5	
	.5	.5	1	...	.5	.5	
				...			
	.5	.5	.5	...	1	.5	
I_{22} :	0	0	1	...	.5	0	3 tuples
	0	0	1	...	.5	.5	corresponding to
	0	0	1	...	.5	1	$\neg x_1 \vee \neg x_2 \vee x_3$
				...			

Figure 4: Construction example.

- $t_{ij}(x) = 0$ if $\chi_{i1} \vee \chi_{i2} \vee \chi_{i3}$ contains the negation of x,
- $t_{ij}(x) = 1$ if $\chi_{i1} \vee \chi_{i2} \vee \chi_{i3}$ contains x non-negated,
- $t_{ij}(r) = j/2$, and
- $t_{ij}(x) = 0.5$ otherwise.

No term of Δ contains both x and the negation of x, is assumed without loss of generality. Let I_{22} be the smallest relation over U containing t_{i0}, t_{i1}, and t_{i2} for every $i \in [1..m]$. Let $I_2 = I_{21} \cup I_{22}$.

Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation with I_1 and I_2 as defined above, and $I_i = \{\}$ if $i > 2$. The construction is illustrated in Figure 4.

Let p be the number of tuple pairs of $I_1 \times I_2$. Let

$$ts = \frac{v}{p} \text{ and } tc = 1.$$

Clearly, $ts > 0$. Let $\tau = V \rightarrow_{NextOne} \{r\}$, a TD template over U. We claim that R is a reduction from 3SAT to TDMINE(D). To prove our claim, we have to establish two things: (1) that any formula Δ has a satisfying truth assignment iff the answer to the TDMINE(D) problem (U, I, τ, ts, tc) is "yes"; and (2) that R can be computed in space $\log m$.

Assume that for some $\Phi \rightarrow_{NextOne} \Psi$ of $[\![\tau]\!]$, we have $I \models^s_c \Phi \rightarrow_{NextOne} \Psi$ with $s \geq ts$ and $c \geq tc$. By Lemma 7, $\Psi(r) \in \{\leq, \neq, \geq\}$ is assumed without loss of generality. We show that Δ has a satisfying truth assignment.

Let k be the number of tuple pairs of $I_1 \times I_{21}$ satisfying Φ. It can be readily seen that $k \leq v$. Obviously, the number of tuple pairs of $I_1 \times I_{22}$ satisfying Φ is a multiple of three—let it be $3n$. Let κ (kappa) be a number such that

$$\kappa = \begin{cases} 0 & \text{if } \Psi(r) = \text{“} \neq \text{”} \\ k & \text{otherwise} \end{cases}$$

Then $I \models^s_c \Phi \rightarrow_{NextOne} \Psi$ with

$$s = \frac{\kappa + 2n}{p} \text{ and } c = \frac{\kappa + 2n}{k + 3n}.$$

Note that κ and n are not both equal to 0 since $s \geq ts > 0$. Then $c \geq tc$ and $s \geq ts$ imply

$$\kappa = k \text{ and } n = 0 \text{ and } k \geq v.$$

Hence, $\Psi(r)$ is either $\leq$ or $\geq$, and $k = v$. One can easily check that $k = v$ implies that $\Phi \rightarrow_{NextOne} \Psi$ belongs to $[\![\tau]\!]^{\{\leq,\geq\}}$.

We now consider the implications of $n = 0$. For example, a term $\neg x_1 \vee \neg x_2 \vee x_3$ in Δ gives rise to a tuple $\{x_1 : 0, x_2 : 0, x_3 : 1, x_4 : 0.5, \ldots, x_v : 0.5, r : 0.5\}$ in I. $n = 0$ implies that $\Phi(x_1) =$ " $\leq$ " or $\Phi(x_2) =$ " $\leq$ " or $\Phi(x_3) =$ " $\geq$ ".

Let B be a truth assignment to the variables of V satisfying ($i \in [1..v]$):

$$B(x_i) = \begin{cases} \textbf{true} & \text{if } \Phi(x_i) = \text{" } \geq \text{"} \\ \textbf{false} & \text{if } \Phi(x_i) = \text{" } \leq \text{"} \end{cases}$$

$n = 0$ implies that B is a truth assignment satisfying Δ.

Conversely, it can now be easily seen that if Δ has a satisfying truth assignment, then for some $\Phi \rightarrow_{NextOne} \Psi \in [\![\tau]\!]$, $I \models^s_c \Phi \rightarrow_{NextOne} \Psi$ with $s \geq ts$ and $c \geq tc$. To see that R can be computed in $\log m$ space, note that $R(\Delta)$ can be written directly from Δ. This concludes the proof. □

Remark: The TD mined in Theorem 5 only uses the operators $\leq$ and $\geq$. This leads to the following corollary.

Corollary 2 *If Θ contains $\leq$ and $\geq$ then* TDMINE(D)$_\Theta$ *is* **NP***-complete.*

PROOF. This follows immediately from the fact that in the proof of Theorem 5, the TD $\Phi \rightarrow_{NextOne} \Psi$ belongs to $[\![\tau]\!]^{\{\leq,\geq\}}$. □

The following theorem states that if one does not consider the "composite" operators $\leq$, $\geq$, and $\neq$ then the resulting TD mining problem is in **P**.

Theorem 6 TDMINE(D)$_{\{<,=,>\}}$ *is in* **P**.

PROOF. Consider the TDMINE(D)$_{\{<,=,>\}}$ problem (U, I, τ, ts, tc) with $\tau = X \rightarrow_\alpha Y$. For every $(i, j) \in \alpha$, for every $(s, t) \in I_i \times I_j$, one can construct in $\mathcal{O}(\mathcal{D})$ time the *unique* TD $\Phi \rightarrow_\alpha \Psi$ of $[\![\tau]\!]^{\{<,=,>\}}$ such that (s, t) satisfies both Φ and Ψ. For each TD so constructed one can compute in polynomial time the support and the confidence, and verify whether these values exceed ts and tc respectively. This concludes the proof. □

6.4 Algorithmic Aspects

In this section, we discuss algorithms to solve certain TDMINE problems. We first give a polynomial time algorithm for TDMINE$_{\{<,=,>\}}$, and then we discuss a significant variant with fairly reduced time requirements. Finally, we discuss the applicability of some existing techniques for increasing the performance of TD mining.

6.4.1 Solving TDMINE$_{\{<,=,>\}}$

Theorem 6 suggests a naive way to solve a TDMINE$_{\{<,=,>\}}$ problem (U, I, τ, ts, tc). We now present a better approach.

Definition 14 Let U be a set of attributes. A *comparator* over U is a DAS Φ over U such that for each $A \in U$, $\Phi(A)$ is either $<$, $=$, or $>$. Let s, t be tuples over U. The *comparator of s with t*, denoted $comp(s, t)$, is the comparator Φ over U such that $\Phi^*(s, t)$. □

For example, if $s = \{A : 0, B : 0\}$ and $t = \{A : 1, B : 0\}$ then $comp(s, t)$ is the DAS $\{(A, <), (B, =)\}$. Comparators correspond to a notion with the same name in [14, page 161]. To solve (U, I, τ, ts, tc) with $\tau = X \rightarrow_\alpha Y$, we proceed in three steps.

step1 For every $(i,j) \in \alpha$, for every $s \in I_i$, for every $t \in I_j$, we store $comp(s[XY], t[XY])$ in a list L. In the worst case, the number of comparators in L is equal to

$$\sum_{i=1}^{\mathcal{N}} (\sum_{j=i}^{\mathcal{N}} |I_i| \cdot |I_j|); \text{ that is, } \mathcal{O}(\mathcal{C}^2).$$

step2 Let $X = \{A_1, \ldots, A_m\}$ and $Y = \{A_{m+1}, \ldots, A_n\}$ $(1 \leq m \leq n)$. The list L of comparators built in step1 is ordered by ascending $A_1, \ldots, A_n$ in $\mathcal{O}(\mathcal{C}^2 \log \mathcal{C})$ time. We order $<$ before $=$, and $=$ before $>$. Figure 5 shows an ordered list of comparators over XY with $X = \{A_1, A_2, A_3\}$ and $Y = \{A_4, A_5, A_6\}$.

step3 Finally, the solution to the TDMINE$_{\{<,=,>\}}$ problem can be computed in a sequential scan of the ordered list of $\mathcal{O}(\mathcal{C}^2)$ comparators. This is illustrated by the procedure *ThresholdTDs* of Figure 6. The procedure delimits "blocks" of comparators that agree on X, and within each such block further delimits blocks of comparators that also agree on Y. For example, from the first seven comparators in Figure 5, it is clear that the confidence of $(A_1,<)(A_2,<)(A_3,<) \rightarrow_\alpha (A_4,<)(A_5,<)(A_6,<)$ equals $3/7$, and the support equals $7/p$, where p is equal to $\sum_{(i,j)\in\alpha} |I_i \times I_j|$.

The above procedure has an overall complexity of $\mathcal{O}(\mathcal{C}^2 \log \mathcal{C})$ in terms of $\mathcal{C}$. It takes linear time in $\mathcal{D}$. As any algorithm that takes $\mathcal{O}(C^2)$ or more time may be very expensive in practical applications, we next derive an interesting variant with reduced time requirements.

6.4.2 Entity Evolution

TDMINE$_\Theta$ requires that the operators appearing in the solution TDs belong to Θ. One could consider specifying the set of allowed operators on an attribute by attribute basis—rather than for the TD as a whole. In particular, for certain attributes, such as `SS#`, equality $(=)$ and inequality $(\neq)$ are often the only meaningful operators. We found that many practical TDs compare primary key attributes for equality. This can be explained as follows. In many applications, tuples represent real-life entities (for example, employee tuples), and primary keys represent identifiers of entities (for example, `SS#`). Often one is interested to see how certain properties (for example, `Sal`) of an entity evolve in time. Tendencies in entity evolution are typically captured by TDs of the form $\Phi \rightarrow_\alpha \Psi$ where $atts(\Phi)$ contains the primary key K of the relation schema under consideration, and $\Phi(A) =$ " $=$ " for each $A \in K$. Most example TDs in this paper have this form, like $(\texttt{SS\#}, =)(\texttt{Rank}, <) \rightarrow_{Next} (\texttt{Sal}, \leq)$

Let $I = \langle I_1, I_2, \ldots \rangle$ be a temporal relation over U. Assume $K \subseteq U$ serves as the primary key. Formally, for each $i \in \mathbb{N}$, for every $s, t \in I_i$, $s[K] = t[K]$ implies $s = t$. Suppose we want to solve a TDMINE$_\Theta$ problem (U, I, τ, ts, tc) where $\tau = KX \rightarrow_\alpha Y$, and that we are only interested in TDs $\Phi \rightarrow_\alpha \Psi$ of $[\![\tau]\!]^{\{<,=,>\}}$ satisfying $\Phi(A) =$ " $=$ " for each $A \in K$. If every I_i is listed in order of ascending primary key, then the problem can be solved as follows.

step1 For every $(i,j) \in \alpha$, for every $s \in I_i$, for every $t \in I_j$, if $s[K] = t[K]$ then we store $comp(s[XY], t[XY])$ in a list L. As I_i and I_j are ordered by K, tuples that agree on K can be found using a merge algorithm. This is illustrated by the procedure *StoreComparators* of Figure 7. This step considers at most

$$\sum_{i=1}^{\mathcal{N}} |I_i| + \sum_{i=1}^{\mathcal{N}-1} (\sum_{j=i+1}^{\mathcal{N}} |I_i| + |I_j|)$$

tuple pairs. It can be proved by simple induction on $\mathcal{N}$ that the latter expression equals $\mathcal{NC}$. Hence, L contains at most $\mathcal{NC}$ comparators.

step2 and step3 The further processing does not differ from the one in Section 6.4.1. The list L of comparators can be sorted in $\mathcal{O}(\mathcal{NC} \log(\mathcal{NC}))$ time, and the solution to the problem under consideration can be computed from the ordered list in $\mathcal{O}(\mathcal{NC})$ time.

	X			Y		
	A_1	A_2	A_3	A_4	A_5	A_6
1	$<$	$<$	$<$	$<$	$<$	$<$
2	$<$	$<$	$<$	$<$	$<$	$<$
3	$<$	$<$	$<$	$<$	$<$	$<$
4	$<$	$<$	$<$	$<$	$>$	$>$
5	$<$	$<$	$<$	$<$	$>$	$>$
6	$<$	$<$	$<$	$=$	$<$	$<$
7	$<$	$<$	$<$	$=$	$<$	$<$
8	$<$	$<$	$=$	$=$	$<$	$<$
...		...			...	

Figure 5: Comparators over XY ordered by $A_1, A_2, \ldots, A_6$.

procedure *ThresholdTDs*
INPUT: Attribute sets X and Y.
List L of comparators over XY, ordered by X, Y.
% L contains $|L|$ comparators;
% $L(i)[X]$ denotes the projection on X of the i^{th} comparator.
Threshold support ts.
Threshold confidence tc.
Support denominator p.
DECLARE: Integer lx, ux, ly, uy.
% Invariant: $lx \leq ly \leq uy \leq ux$.
Real s, c.
Comparator Φ over X.
Comparator Ψ over Y.
begin
$lx = 1$; $ux = 1$
while $ux \leq |L|$
  $\Phi = L(lx)[X]$
  while $ux \leq |L|$ and $L(ux)[X] = \Phi$ **loop** $ux = ux + 1$ **endloop**
  $ly = lx$; $uy = lx$
  while $uy < ux$ **loop**
    $\Psi = L(ly)[Y]$
    while $uy < ux$ and $L(uy)[Y] = \Psi$ **loop** $uy = uy + 1$ **endloop**
    % Every comparator $L(i), lx \leq i < ux$, contains Φ.
    % Every comparator $L(j), ly \leq j < uy$, contains Ψ.
    % Next we compute the support and confidence of $\Phi \rightarrow_\alpha \Psi$.
    $s = (uy - ly)/p$
    $c = (uy - ly)/(ux - lx)$
    if $s \geq ts$ and $c \geq tc$ **then output** $\Phi \rightarrow_\alpha \Psi$ **endif**
    $ly = uy$
  endloop
  $lx = ux$
endloop
end

Figure 6: Procedure *ThresholdTDs*.

```
procedure StoreComparators
INPUT:    Attribute set U.
          Attribute sets K, X, Y ⊆ U.
          % A temporal relation is represented by a list of 𝒩 "snapshot" relations;
          % each "snapshot" relation I_i, 1 ≤ i ≤ 𝒩, is a list of tuples over U.
          % Each I_i contains |I_i| tuples. I_i(k) denotes the k^th tuple of I_i.
          Temporal relation ⟨I_1, ..., I_𝒩⟩ over U with
              for each i ∈ [1..𝒩], for each k, l ∈ [1..|I_i|] (let s = I_i(k) and t = I_i(l)),
              k < l implies s[K] < t[K]—i.e., I_i is ordered by key K.
          TAR α with for each (i, j) ∈ α, j ≤ 𝒩.
DECLARE:  Integer cnt.
          List L of comparators over XY.
          Tuples s, t over U.
begin
cnt = 0
for each (i, j) ∈ α loop
    if i = j then
        % Every tuple of I_i is compared with itself.
        for each k ∈ [1..|I_i|] loop
            cnt = cnt + 1
            L(cnt) = the comparator Φ over XY with Φ(A) = " = " for each A ∈ XY
        endloop
    else % i ≠ j
        % The following loop resembles a "merge" of I_k and I_l.
        k = 1; l = 1
        while k ≤ |I_i| and l ≤ |I_j| loop
            s = I_i(k); t = I_j(l)
            if s[K] < t[K] then k = k + 1
            elsif s[K] > t[K] then l = l + 1
            else % s[K] = t[K]
                cnt = cnt + 1
                L(cnt) = comp(s[XY], t[XY])
            endif
        endloop
    endif
endloop
return L
end
```

Figure 7: Procedure *StoreComparators*.

I_1 :

SS#	Rank	Sal
A1	1	100
B2	1	120
C3	3	140
D4	2	80
E5	2	120

I_2 :

SS#	Rank	Sal
A1	2	110
B2	2	110
C3	2	130
D4	3	90
E5	3	120

Comparators:

Rank	Sal
<	<
<	>
>	>
<	<
<	=

Ordered:

	Rank	Sal
1	<	<
2	<	<
3	<	=
4	<	>
5	>	>

Figure 8: Entity Evolution.

The above procedure is linear in $\mathcal{D}$. It has a worst-case complexity of $\mathcal{O}(\mathcal{NC}\log(\mathcal{NC}))$. We can think of practical applications where $\mathcal{N}$ is relatively small compared to $\mathcal{C}$, and the complexity may become acceptable. For example, a typical medical experiment may collect daily blood pressure readings from 100 patients during a one year period. The resulting databases has $\mathcal{N} = 365$ and $\mathcal{C} = 36,500$. We note that in many situations, the TAR involved in a TDMINE problem will not require comparing every timeslice with every other timeslice. For example, if the TAR involved is *Next* then the list L of comparators constructed in step1 will contain not more than $2\mathcal{C}$ comparators (instead of $\mathcal{NC}$).

Example 6 The algorithm is further illustrated by Figure 8. We start from a temporal relation $I = \langle I_1, I_2, \ldots \rangle$ over $\{\texttt{SS\#}, \texttt{Rank}, \texttt{Sal}\}$. Suppose $I_i = \{\}$ if $i > 2$; that is, $\mathcal{N} = 2$. We are interested in mining TDs of the form $(\texttt{SS\#}, =)(\texttt{Rank}, \theta_1) \rightarrow_{NextOne} (\texttt{Sal}, \theta_2)$ where θ_1 and θ_2 are operators of $\{<, =, >\}$. The figure shows the list of comparators computed in step1, and the ordered list computed in step2. From the ordered list, it can be readily seen, for example, that the TD $(\texttt{SS\#}, =)(\texttt{Rank}, <) \rightarrow_{NextOne} (\texttt{Sal}, <)$ has a confidence of 2/4. □

6.4.3 Other Techniques

TDMINE can be solved in polynomial time when time requirements are expressed as a function of $\mathcal{C}$. If the input of TDMINE is characterized by $\mathcal{D}$, then it can be solved in polynomial time only if **P=NP**. Nevertheless, since $\mathcal{C}$ is generally in the order of thousands or millions, the cardinality may still be of overriding importance compared to the degree. Techniques that have been successfully applied in mining other rules, may also be applicable to TDs:

- Tuple reduction by generalization [17] or sampling [13].
- Attribute reduction techniques [19].
- Incremental maintenance of mined rules [7].
- Visualization.

Tuple reduction by sampling can be done before starting TD mining. This provides no way to circumvent the **NP**-completeness result shown earlier, but increases performance by reducing $\mathcal{C}$. Attribute reduction and incremental maintenance are fairly new and deserve further investigation. We conclude this section by a note on visualization.

Visualization techniques are generally considered a useful method for discovering patterns in data sets. This, of course, relies on some user intervention. The problem here is that it is difficult to visualize data spaces with high dimensionality. Every attribute appearing in a TD is a dimension. Moreover, the time dimension is inherent in every TD. The practicality of visualization therefore depends on the possibility to solve a given TDMINE problem by only dealing with "short" TDs at a time.

So a question of practical importance is: Can a given TDMINE problem involving $\mathcal{D}$ attributes be solved by (a) first *decomposing* it in polynomial time into a number of TDMINE subproblems of a specified visualizable dimension (practically 2 or 3), (b) then solving the smaller subproblems using visualization techniques, and (c) finally merging the solutions in polynomial time so as to obtain a solution for the original TDMINE problem? It is of interest to note immediately that "shortening" the left-hand DAS of a TD can result in an increase as well as a decrease of the confidence. Recall that the confidence c of a TD $\Phi \rightarrow_\alpha \Psi$ is equal to a proportion b/l where b is the number of tuple pairs satisfying both Φ and Ψ, and l is the number of tuple pairs satisfying Φ. Replacing Φ by a proper subset of Φ will result in both b and l non-decreasing. So c can increase as well as decrease. More fundamentally, we showed that TDMINE is **NP**-complete if the input is characterized by the number of attributes. In the suggested decomposition strategy, all subproblems are of the same specified dimension and hence the time required to solve any one subproblem is irrespective of $\mathcal{D}$. But then the decomposition strategy corresponds to a polynomial-time algorithm, which exists only if **P=NP**. This is a strong indication that there is no such decomposition strategy.

7 Comparison with Related Work

In this section, we compare our work with related studies. We first show that our time model can express Wang et al.'s *temporal types* [29]. Then we argue that TDs can capture most temporal dependencies found in the literature. Finally, we compare TD mining with work on the mining of association rules.

7.1 Temporal Types

The construct of *temporal type* [29] serves to model time granularity and is a generalization of several earlier proposals in the literature. We compare our notion of TAR with the construct of temporal type. In particular, we indicate that temporal types correspond to the restricted class of TARs which we called chronologies.

Definition 15 [29] A *temporal type* is a mapping μ from $\mathbb{N}$ to $\wp(\mathbb{R})$ such that for every $i, j \in \mathbb{N}$,

1. if $i < j$ then every real number of $\mu(i)$ is strictly less than every real number of $\mu(j)$, and
2. if $\mu(i) = \{\}$ and $i \leq j$ then $\mu(j) = \{\}$.

$\mathbb{N}$ is called the *index set*, and $\mathbb{R}$ the *absolute time set*. □

We now establish a one-to-one mapping between chronologies and temporal types. In particular, we define an injective function mapping every chronology α to a temporal type, denoted $\alpha^{\mathbf{tt}}$.

Definition 16 Let α be a chronology and let $P_1, P_2, P_3, \ldots$ be the equivalence classes of $\alpha^{\mathbf{sym}}$. Let n be the number of equivalence classes (possibly $n = \infty$). By the definition of chronology (Definition 3), $P_1 \prec P_2 \prec P_3 \prec \ldots$ is assumed without loss of generality. From α, we derive a mapping $\alpha^{\mathbf{tt}}$ from $\mathbb{N}$ to $\wp(\mathbb{N})$ as follows:

- $\alpha^{\mathbf{tt}}(i) = P_i$ if $i \leq n$, and
- $\alpha^{\mathbf{tt}}(i) = \{\}$ if $i > n$.

□

For example, consider the TAR `Month` introduced in Section 2.2. Then $\texttt{Month}^{\mathbf{tt}}(1) = \{1, \ldots, 31\}$, $\texttt{Month}^{\mathbf{tt}}(2) = \{32, \ldots, 59\}$,..., $\texttt{Month}^{\mathbf{tt}}(13) = \{366, \ldots, 396\}$, and so on. It can be easily seen that $\alpha^{\mathbf{tt}}$ *is* a temporal type. Note incidentally that there are temporal types that cannot be expressed as the result of applying the above transformation to a chronology. This is only because temporal types are functions into $\wp(\mathbb{R})$, while $\alpha^{\mathbf{tt}}$ is a function into $\wp(\mathbb{N})$. The concept of temporal type does not critically depend, however, on $\mathbb{R}$ being the absolute time set. Wang et al. [29, page 123] mention that "in fact, any infinite set with a total ordering can serve as the absolute time; reals, rationals and integers are examples." Moreover, Wang et al. [29] do not rely on properties that hold for $\mathbb{R}$ but not for $\mathbb{N}$, such as density.

Wang et al. [29] define a relation *is-finer-than* on the family of temporal types. Importantly, an interesting and convenient property of the construct of chronology is that set inclusion ($\subseteq$) captures the meaning of *is-finer-than*. For example, one would typically have $\texttt{Month} \subseteq \texttt{Year}$—a month falls entirely within a single year—but $\texttt{Week} \not\subseteq \texttt{Month}$—a new month can start in the middle of a week. This is an interesting property, as well-known properties of set inclusion can be readily used in reasoning about chronologies. For example, it is relatively easy to show that the set of all chronologies, ordered by inclusion, is a lattice.

Bettini et al. [6] give more general definitions of temporal types. In particular, they allow the index set to be any chain isomorphic to a subset of the integers endowed with its usual order, and they allow the absolute time set to be any chain. Also, they consider relaxations of condition (2) in Definition 15. Such extensions are beyond the scope of TARs.

To conclude, chronologies correspond to temporal types. Importantly, there are many meaningful TARs, for example *Next*, which are non-chronological, and which have not been explicitly captured by previous time models.

7.2 Database Constraints

Temporal Dependencies

Lately, there has been a growing interest in dependencies for temporal databases [20, 29]. All temporal dependencies found in the extensive overview of Jensen et al. [20] compare attributes for equality (=) only. It seems that TDs are among the first temporal dependencies that introduce operators other than equality.

Our treatment of TDs is related to the work on temporal database design of Wang et al. [29]. In Section 7.1, we already compared our time model with theirs. Wang et al.'s *temporal* FDs (TFDs) correspond to a restricted class of TDs where, first, equality is the only operator used, and second, the TAR involved is a chronology. An example is $(\texttt{SS\#}, =) \rightarrow_{\texttt{Month}} (\texttt{Sal}, =)$.

Jensen et al. [20] essentially extend the notion of satisfaction of FDs for temporal relations. Jensen et al.'s TFDs correspond to TDs where, first, equality is the only operator used, and second, the TAR involved is *Current*. An example is $(\texttt{SS\#}, =) \rightarrow_{Current} (\texttt{Sal}, =)$.

In the work of Vianu [28], a temporal relation is viewed as a sequence of snapshot relations in time—a view that is also present in our notion of temporal relation. Tuples preserve their identity through time. The dynamic constraints used in conjunction with the data model are restricted to certain analogs of FDs, called *dynamic* FDs (DFDs). For example, consider a company-wide salary update between times i and $i+1$. Each new salary is determined strictly on the basis of the old salary and rank. Hence, two employees with the same salary and rank receive the same new salary as a result of the salary update. For each attribute A, let $\overset{\vee}{A}$ and $\hat{A}$ refer to A-values before and after an update respectively. The constraint pertaining to the salary update is expressed by the DFD:

$$\{\overset{\vee}{\texttt{Sal}}, \overset{\vee}{\texttt{Rank}}\} \rightarrow \{\overset{\wedge}{\texttt{Sal}}\}$$

Two tuples with the same value for `Sal` and `Rank` before the update must agree on `Sal` after the update. It can be shown that Vianu's DFDs cannot be expressed by TDs, and vice versa.

Navathe and Ahmed [22, 23] introduce a temporal dependency to express that certain attribute values always change simultaneously. This notion of synchronism can be expressed in terms of TDs, and hence is less general. For example, the TDs $(\texttt{SS\#}, =)(\texttt{Rank}, \neq) \rightarrow_{Next} (\texttt{Sal}, \neq)$ and $(\texttt{SS\#}, =)(\texttt{Sal}, \neq) \rightarrow_{Next} (\texttt{Rank}, \neq)$ taken together express that an employee's rank and salary always change together.

TDs evolved from the *dynamic* FDs (DFDs) and the *temporal* FDs (TFDs) of Wijsen [31, 30], by introducing TARs and by allowing operators other than equality. An extension of TFDs with object-identity is proposed in [33].

Finally it should be mentioned that several researchers have used first-order temporal logic (FOTL) to express temporal database integrity; among others, Chomicki [8], Chomicki and Niwiński [9], Lipeck and Saake [21].

Non-Temporal Dependencies

For non-temporal relational databases, constraints involving order have been proposed in the literature. The *order dependencies* proposed by Ginsburg and Hull [14, 15] generalize FDs by comparing attributes not only for equality (=), but also for order ($\{<, >, \leq, \geq\}$). Inequality ($\neq$) is not considered. The order is not required to be total. The major contribution of TDs compared with order dependencies is the explicit modeling of the time dimension in TDs.

Constraint-generating dependencies proposed by Baudinet et al. [5] generalize *equality-generating dependencies*, which subsume FDs, by replacing equality requirements by constraints on an interpreted domain. A special type of constraint-generating dependency fixes the language of constraint formulas to equality (=), inequality ($\neq$), and order ($<, \leq$) constraints. Theorem 2 is directly based on a result of [5].

In a recent work, Guo et al. [16] study satisfiability and implication problems for sets of inequalities of the form $X\theta Y$ and $X\theta c$, where X and Y are variables, c is a constant, and θ is an operator of $\{<, =, >, \leq, \geq, \neq\}$.

7.3 Data Mining

Recently there has been a growing interest in the mining of different types of *association rules* from large relational tables. Association rules can take different forms [3, 18, 26]. Most work in association mining has concentrated on discovering rules of the form

$$\forall t[(R(t) \wedge C(t)) \rightarrow C'(t)]$$

where C and C' are constraint formulas relating certain attribute values of the tuple t with specified constants. An example is the rule (taken from [26]) *"Married employees between 30 and 49 years old have two cars,"* which can be expressed as

$$\forall t\{[emp(t) \wedge 30 \leq t(\texttt{Age}) \leq 49 \wedge t(\texttt{Married}) = \text{Yes}] \rightarrow t(\texttt{NumCars}) = 2\},$$

and which is commonly abbreviated to

$$(\texttt{Age} : 30..49) \text{ and } (\texttt{Married} : \text{Yes}) \Rightarrow (\texttt{NumCars} : 2).$$

The support s of an association rule is the percentage of tuples satisfying both the left-hand and the right-hand side of the rule. The confidence is c if c% of the tuples satisfying the left-hand side of the rule also satisfy the right-hand side. Clearly, our notions of support and confidence not only have the same name, but also the same set-up and intention. Certain studies limit the length of a rule to enable *visualization* [12, 13].

Note that the association rules just mentioned compare certain attribute values of a tuple with specified constants. Each individual tuple can give evidence for or against the association. The TDs proposed in this paper compare attributes in one tuple with the corresponding attributes in another tuple. That is, TD satisfaction is expressed in terms of tuple pairs—rather than individual tuples. Following the terminology of Baudinet et al. [5], TDs are constraint-generating 2-dependencies, whereas classical association rules are constraint-generating 1-dependencies.

In this study, we assumed a single temporal relation. Other researchers have also considered the problem of mining rules from relations that are built from multiple base relations using some query language [4].

Most work on data mining concerns in the first place the performance of algorithms. Examples are [2, 11, 24, 25]. In this study, we have proceeded in a different way and have started with analyzing the complexity of the TDMINE problem itself—rather than algorithms to solve it. The rationale behind this approach is that complexity analysis gives us important indications about the tractability of the problem in hand, which may complement algorithm design techniques.

8 Summary

We introduced *trend dependencies* (TDs), which allow expressing significant temporal trends. The time dimension is captured by TDs through the concept of *time accessibility relation* (TAR). We showed that TARs can express time granularities in a simple and elegant way. We provided a characterization of the satisfiability problem, and we showed it is the dual of the logical implication problem. The satisfiability problem for TDs turns out **NP**-hard. An upper bound for the complexity depends on the formalism used to represent TARs. We provided an axiomatization for reasoning about TDs.

As TDs allow capturing significant knowledge, mining them from existing databases is interesting and important. We studied the problem TDMINE: Given a temporal database, mine the TDs of a specified TD class whose support and confidence exceed specified minimum thresholds. Time requirements were expressed in terms of the cardinality $\mathcal{C}$ and the number of attributes $\mathcal{D}$. We showed that TDMINE(D) is **NP**-complete if time requirements are expressed as a function of $\mathcal{D}$. This implies that TDMINE cannot be "scaled down" to lower dimensions, which limits the practicality of visualization techniques for discovering TDs among many attributes. Although solving TDMINE is generally expensive, we worked out an interesting variant with acceptable time requirements.

We showed that TDs can express several temporal dependencies found in the literature. TDs seem to be among the first temporal dependencies that compare attributes using operators other than equality.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] A. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, 1994.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D.C., 1993.

[4] E. Baralis and G. Psaila. Designing templates for mining association rules. *Journal of Intelligent Information Systems*, 9(1):7–32, 1997.

[5] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. In *Proc. 5th Int. Conf. on Database Theory*, LNCS 893, pages 322–337. Springer-Verlag, 1995.

[6] C. Bettini, X. Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 68–78, Montreal, Canada, June 1996. ACM Press.

[7] D. W. Cheung, J. Han, V. T. Ng, and C. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Int. Conf. Data Engineering*, pages 106–114, New Orleans, Louisiana, 1996.

[8] J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Trans. on Database Systems*, 20(2):148–186, June 1995.

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

[10] J. Clifford, A. Crocker, and A. Tuzhilin. On completeness of historical relational query languages. *ACM Trans. on Database Systems*, 19(1):64–116, 1994.

[11] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 163–174, San Jose, CA, 1995.

[12] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 13–23, Montreal, Canada, 1996.

[13] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 182–191, Montreal, Canada, 1996.

[14] S. Ginsburg and R. Hull. Order dependency in the relational model. *Theoretical Computer Science*, 26:149–195, 1983.

[15] S. Ginsburg and R. Hull. Sort sets in the relational model. *Journal of the ACM*, 33:465–488, 1986.

[16] S. Guo, W. Sun, and M. Weiss. Solving satisfiability and implication problems in database systems. *ACM Trans. on Database Systems*, 21(2):270–293, 1996.

[17] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):29–40, 1993.

[18] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. Int. Conf. Very Large Data Bases*, pages 420–431, Zürich, Switzerland, 1995.

[19] X. Hu and N. Cercone. Mining knowledge rules from databases: A rough set approach. In *Int. Conf. Data Engineering*, pages 96–105, New Orleans, Louisiana, 1996.

[20] C. Jensen, R. Snodgrass, and M. Soo. Extending existing dependency theory to temporal databases. *IEEE Trans. on Knowledge and Data Engineering*, 8(4):563–582, 1996.

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

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

[23] S. Navathe and R. Ahmed. Temporal extensions to the relational model and SQL. In A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*, chapter 4, pages 92–109. Benjamin/Cummings, 1993.

[24] J. Park, M.-S. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 175–186, San Jose, CA, 1995.

[25] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. Int. Conf. Very Large Data Bases*, pages 432–443, Zürich, Switzerland, 1995.

[26] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 1–12, Montreal, Canada, 1996.

[27] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation.* Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA, 1993.

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

[29] X. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical design for temporal databases with multiple granularities. *ACM Trans. on Database Systems*, 22(2):115–170, 1997.

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

[31] J. Wijsen. *Extending Dependency Theory for Temporal Databases.* PhD thesis, Katholieke Universiteit Leuven, Belgium, Feb. 1995.

[32] J. Wijsen. Reasoning about qualitative trends in databases. *Information Systems*, 23(7):469–493, 1998.

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

[34] S. Yen and A. Chen. The analysis of relationships in databases for rule derivation. *Journal of Intelligent Information Systems*, 7(3):235–259, 1996.