Trends in Databases: Reasoning and Mining

Jef Wijsen

Abstract—We propose a temporal dependency, called trend dependency (TD), which captures a significant family of data evolution regularities. An example of such regularity is "Salaries of employees generally 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. As TDs allow expressing meaningful trends, "mining" them from existing databases is interesting. For the purpose of TD mining, TD satisfaction is characterized by support and confidence measures. 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 the problem.

Index Terms—Temporal database, knowledge discovery, data mining, functional dependency.

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) [1], [2]. 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. [3]. In this paper, we introduce a 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"* [1] and *"A faculty's rank cannot change during an academic year"* [4].

Database integrity is probably the 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. 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 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.

• *The author is with the University of Mons-Hainaut, Belgium. E-mail: Jef.Wijsen@umh.ac.be.*

Manuscript received 2 June 1997; revised 12 Jan. 1998; accepted 2 Aug. 1999. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 104006.

Checking temporal constraints typically involves comparing tuples that are valid at different time instances. 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 $\{(i, i+1) \mid i \in \mathbb{N}\}$, called $Next$. The meaning of TDs can now be easily explained by a formula 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 is then expressed by the universal closure of the formula:

$$[\ emp(s,i) \wedge emp(t,j) \wedge Next(i,j) \wedge s(\texttt{SS\#}) = t(\texttt{SS\#})\ ] \rightarrow s(\texttt{Sal}) \leq t(\texttt{Sal}),$$

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 express several temporal dependencies found in the literature. They seem to be among the first temporal dependencies that compare attributes by operators other than equality. In this paper, we 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 "nontrivial" 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?"

The latter problem 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? The strength of a trend will be characterized by extending the common measures of *support* and *confidence* [5].

Another interesting question concerns the axiomatizability of TDs: how to find a sound and complete set of inference rules for reasoning about TDs. Such an axiomatization has already been provided in [6].

The outline of the paper is as follows: Section 2 introduces two components of TDs, namely, *directed attribute sets* (DAS) for comparing tuples, and *time accessibility relations* (TAR) for relating time points. Section 3 then formalizes the concept of TD. Section 4 concerns the complexity of the logical implication problem and the satisfiability problem. Section 5 concerns mining TDs from temporal databases. The problem TDMINE is defined and its complexity as well as algorithmic aspects are studied. A preliminary version of this section first appeared in [7]. A comparison with related work is contained in Section 6. Finally, Section 7 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 disequality ($\neq$). Attributes like `SS#` fit in our framework as there is no problem in using only equality and disequality 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 compare distinct attributes, a single domain can be assumed without loss of generality.*

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

Fig. 1. Operators of $\mathbf{OP}$ with shorthand and inverse.

The set $\mathbf{OP}$ *is defined as the powerset of* $\{<,=,>\}$. *That is,* $\mathbf{OP} = \wp(\{<,=,>\})$. *Elements of* $\mathbf{OP}$ *are called* operators. *The symbol* θ *will be used to denote operators. We introduce convenient syntactic shorthands for the elements of* $\mathbf{OP}$, *as shown in Fig. 1. Let* $\theta \in \mathbf{OP}$. *The* inverse *of* θ, *denoted* $\overline{\theta}$, *is equal to* $\{<,=,>\} \setminus \theta$.

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* $\mathbf{OP}$. *The symbols* $\Phi, \Psi, \Upsilon, \Omega$ *will be used to denote a DAS.*

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* $\emptyset$. Φ *is called* reflexive *iff for every* $A \in atts(\Phi)$, $\Phi(A)$ *contains the equality* $(=)$ *operator.*

The DAS $\{(A_1,\theta_1),(A_2,\theta_2),\ldots,(A_n,\theta_n)\}$ *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* $s(A)\ \theta_A\ t(A)$ *for each* $A \in X$, *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 $(\texttt{SS\#}, =)(\texttt{Rank}, \leq)(\texttt{Sal}, <)$. 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 the operator "$\wedge$" on DASs is commutative, associative, and idempotent. The following lemma states some properties that will be used later on:

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 section, we first define the construct of TAR and then we show how TARs can be used to model time granularities.

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

2.2.1 Time Accessibility Relations (TARs)

In this work, the time line is represented by the set of natural numbers $\mathbb{N}$. Trends typically compare attributes of one tuple s with attributes of another tuple t, where s and t may belong to different "snapshots." The concept of TAR is used to indicate which tuples have to be compared with one another.

Definition 2. *We define:*

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

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:*

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

The symbols α, β will be used to denote TARs. The complement *of a TAR α, denoted $\neg\alpha$, is defined as $\neg\alpha = Future \setminus \alpha$.*

Note 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 be 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. They 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, etc. Then, the notion of "month" can be captured by the smallest TAR containing the pair $(i,j) \in Future$ whenever i and j represent days of the same month. That is,

$$\texttt{Month} = \{\overbrace{(1,1),(1,2),\ldots,(30,31),(31,31)}^{January,\ 1950},$$
$$\overbrace{(32,32),(32,33),\ldots,(58,59),(59,59)}^{February,\ 1950},\ldots\}.$$

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.

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. Second, months are not "interleaved." A TAR satisfying these properties is 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 $i < j$ for every $i \in P$ and $j \in Q$. 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. The TAR α is called a chronology *iff*

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

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 (part of) the time line in successive time granules. Interestingly, set inclusion ($\subseteq$) on chronologies captures the notion of *finer-than* among time granularities. For example, $\texttt{Month} \subseteq \texttt{Year}$ expresses that every month falls entirely within a single year; but $\texttt{Week} \not\subseteq \texttt{Month}$ as months do not divide evenly into weeks.

The construct of chronology corresponds to the notion of *temporal type* used in [4]. The major difference is that a temporal type explicitly adds a mapping from $\mathbb{N}$ to subsequent equivalence classes. A thorough comparison of chronologies and temporal types can be found in [8].

3 Trend Dependency (TD)

Having defined the notions of DAS and TAR, we are now ready to introduce the concept of TD. 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 particular 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

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

TDs encompass the temporal FDs (TFDs) proposed by Wang et al. [4]. For example, *"An employee cannot have two distinct salaries within the same month,"* is expressed by

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

where Month is defined as in Section 2.2. 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. This 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 $\{\texttt{Patient}, \texttt{Diast}, \texttt{Syst}\}$, storing blood pressure readings from patients. A tuple $\{\texttt{Patient} : x, \texttt{Diast} : y, \texttt{Syst} : z\}$ 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:

$$(\texttt{Patient}, =)(\texttt{Diast}, <) \rightarrow_{Next} (\texttt{Syst}, <).$$

Note that TDs do not allow comparing different attributes with one another. For example, let the inequality $\texttt{Diast} < \texttt{Syst}$ 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 nontrivial issues. For example, the (nonrealistic) TDs $(\texttt{Patient}, =) \rightarrow_{Next} (\texttt{Diast}, <)$ and $(\texttt{Patient}, =) \rightarrow_{Next} (\texttt{Syst}, >)$ taken together express that the diastolic and systolic blood pressure of a patient converge. Then, in order to satisfy $\texttt{Diast} < \texttt{Syst}$, 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 $U \subseteq$ **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 such that for some $n \in \mathbb{N}$, $I_i = \{\}$ for every $i > n$. 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 denote the TD $\Phi \rightarrow_\alpha \Psi$. We call Φ the left-hand *DAS of σ, and Ψ the* right-hand *DAS; we say that the TD* 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. A similar approach is taken in, for example, [1], [9]. Also, the temporal dependency theory of Jensen et al. [3] departs from the idea that temporal relations can be "timesliced." Of course, more advanced structures for storing time-related data have been proposed [10], [11]. However, such representation issues are somehow peripheral to this study. Intuitively, one may think of our temporal relations as the result of *timeslicing* a more 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 "nontrivial" way.

A precise formulation of the latter problem will be given later on. For now, note that every set of TDs is trivially satisfied by the empty temporal relation.

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 that 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 with cardinality one or two. 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 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 $(i, j) \in Future$. 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.[3] 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 (say $I = \langle I_1, I_2, \ldots \rangle$) that satisfies Σ

3. 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 $|I| < |J|$.

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 (say 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., $c, d \in \mathbf{dom}$) with $c < d$. Let $(i,j) \in Future$. 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 for solving TDSAT. For now, we just assume the existence of such a representation. The next section 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.

Example 2. Suppose we want to decide whether a set Σ of TDs logically implies the TD $(\mathtt{SS\#}, =) \rightarrow_{Next} (\mathtt{Sal}, \leq)$ (call it σ_1). Let σ_2 be $\emptyset \rightarrow_{Next} (\mathtt{SS\#}, =)(\mathtt{Sal}, >)$. 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 $s(\mathtt{SS\#}) = t(\mathtt{SS\#})$ and $s(\mathtt{Sal}) > \mathtt{t}(\mathtt{Sal})$. 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 $\Sigma' = \Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \bar{\theta})\}$.*

Proof. Note that the proof is trivial if $\Psi = \emptyset$.

$\Rightarrow$. Assume for some $(A, \theta) \in \Psi$, $\Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \bar{\theta})\}$ 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 $\Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \bar{\theta})\}$. 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 $\Phi^*(s,t)$, but not $\Psi^*(s,t)$. Hence, for some $(A, \theta) \in \Psi$, $[i \triangleright s, j \triangleright t]$ satisfies $\Sigma \cup \{\emptyset \rightarrow_\alpha \Phi \wedge (A, \bar{\theta})\}$. 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 $\Sigma \not\models \emptyset \rightarrow_\alpha (A, \perp)$.*

Proof. By Theorem 1, $\Sigma \not\models \emptyset \rightarrow_\alpha (A, \perp)$ iff $\Sigma \cup \{\emptyset \rightarrow_\alpha (A, \top)\}$ is α-satisfiable. Since $\emptyset \rightarrow_\alpha (A, \perp)$ is a trivial TD, we have $\Sigma \not\models \emptyset \rightarrow_\alpha (A, \top)$ 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. The logical implication problem is **coNP**-complete for this type of dependencies [12]. This concludes the proof. □

As an immediate corollary of Theorems 1 and 2, TDSAT is **NP**-hard. We are now going to 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 $\mathbb{N}$. The relationship between the complexity of TDSAT and the formalism chosen to represent TARs is studied next.

First, we note that certain TDs can only be satisfied by the empty temporal relation. An example is $(\mathtt{SS\#}, =) \rightarrow_{Current} (\mathtt{Sal}, <)$, stating that an employee earns less than his/her salary.

Definition 7. *Let Σ be a set of TDs over the set U of attributes. A natural number $i \in \mathbb{N}$ is called* improper *with regard to 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 *with regard to Σ. We write $prop(\Sigma)$ for the smallest set containing $(i,j) \in Future$ whenever i and j are proper with regard to Σ. Σ is called* consistent *iff $prop(\Sigma) = Future$; otherwise, it is called* inconsistent.

It follows that the singleton set $\{(\texttt{SS\#},=) \rightarrow_{Current} (\texttt{Sal},<)\}$ is inconsistent (every natural number is inconsistent with regard to this set). A similar notion of impropriety is used in [13]. The following lemma states that an inconsistent set of TDs requires that certain timeslices 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 with regard to Σ iff $I_i = \{\}$ for every temporal relation $I = \langle I_1, I_2, \ldots\rangle$ over U that satisfies Σ.*

Proof. $\Rightarrow$. Let $i \in \mathbb{N}$. Assume i is improper with regard to 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 (say 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 with regard to 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 shown next. 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 *with regard to 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 $\{\texttt{neg}, \texttt{pos}\}$. Obviously, $|\mathcal{F}_n| = 2^{n+1}$.

Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a list *of TDs.*[4] *Let $f \in \mathcal{F}_n$. The TAR* induced by *f and Σ, denoted as $[\![f]\!]_\Sigma$, is defined as follows:*

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

where

- *$\beta_0 = Current$ if $f(0) = \texttt{pos}$ and $\beta_0 = \neg Current$ if $f(0) = \texttt{neg}$ and*[5]
- *for every $i \in [1..n]$, $\beta_i = tar(\sigma_i)$ if $f(i) = \texttt{pos}$ and $\beta_i = \neg\, tar(\sigma_i)$ if $f(i) = \texttt{neg}$.*

The concepts of homogeneity and induced TAR are illustrated by Example 3. 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 with regard to Σ.

Example 3. Let Σ be the set $\{\sigma_1, \sigma_2\}$, 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, among others $\{(0,\texttt{neg}), (1,\texttt{pos}), (2,\texttt{neg})\}$ (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 with regard to 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 with regard to 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 with regard to 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

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

5. Recall from Definition 2 that $\neg\alpha = Future \setminus \alpha$.

$(k,l) \in \llbracket f \rrbracket_\Sigma$, $[k \triangleright s, l \triangleright t]$ falsifies Σ. Since s and t are arbitrary, it is correct to conclude that $\llbracket f \rrbracket_\Sigma$ is homogeneous with regard to Σ. It can be readily seen that testing Conditions 1, 2, and 3 is independent of the size of $\llbracket f \rrbracket_\Sigma$. 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 $(U, \Sigma, \alpha \cap \llbracket f \rrbracket_\Sigma)$ is "yes" for some $f \in \mathcal{F}_n$. By Lemmas 6 and 3, a *nondeterministic* 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* $(k,l) \in \llbracket f \rrbracket_\Sigma$, and, hence, for all $(k,l) \in \llbracket f \rrbracket_\Sigma \cap \alpha$. But, remark: Even if this algorithm successfully terminates by finding some s, t, and f such that $[k \triangleright s, l \triangleright t]$ satisfies Σ for all $(k,l) \in \llbracket f \rrbracket_\Sigma \cap \alpha$, then this does *not* imply that the answer to (U, Σ, α) is "yes." This is because we still have to determine whether $\llbracket f \rrbracket_\Sigma \cap \alpha$ is nonempty. 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 the nonemptiness of $\llbracket f \rrbracket_\Sigma \cap \alpha$. The following theorem provides 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 $\llbracket f \rrbracket_\Sigma \cap \alpha$ is nonempty. If TARINTERSECT is in* **P**, *then TDSAT is in* **NP**.

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 $(U, \Sigma, \alpha \cap \llbracket f \rrbracket_\Sigma)$ is "yes" for some $f \in \mathcal{F}_n$. By Lemmas 6 and 3, a *nondeterministic* polynomial algorithm can guess $f \in \mathcal{F}_n$ as well as tuples s and t, and determine in polynomial time whether $\llbracket f \rrbracket_\Sigma \cap \alpha$ is nonempty (an instance of TARINTERSECT), and, if so, whether $[k \triangleright s, l \triangleright t]$ satisfies Σ for *all* $(k,l) \in \llbracket f \rrbracket_\Sigma$. □

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 days of the same month must belong to the same year).

5 TD Mining

As TDs allow expressing significant knowledge about the data stored in a database, discovering them from existing databases is interesting and important. 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 section contains an introductory example. In Section 5.2, we give a formal definition of the data mining problem TDMINE. Its complexity is studied in Section 5.3, and algorithmic aspects are discussed in Section 5.4.

I_1 :			I_2 :		
SS#	Rank	Sal	SS#	Rank	Sal
A1	1	100	A1	2	110
B2	1	120	B2	2	110
C3	3	140	C3	2	130
D4	2	80	D4	3	90
E5	2	120	E5	3	120

Fig. 2. Example database.

5.1 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 gradual notion of satisfaction. For the purpose of TD mining, we are going to characterize TD satisfaction by the notions of *support* and *confidence*, which are introduced in [5] and commonly used in the work on association rule mining. Consider the temporal relation $I = \langle I_1, I_2, \ldots \rangle$ shown in Fig. 2; 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.

5.2 The TD Mining Problem

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

Definition 9. *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 $B_{(i,j)} = \{(s,t) \in I_i \times I_j \mid \Phi^*(s,t)$ and $\Psi^*(s,t)\}$. Let $p = \sum_{(i,j)\in\alpha} |I_i \times I_j|$, $l = \sum_{(i,j)\in\alpha} |L_{(i,j)}|$, and $b = \sum_{(i,j)\in\alpha} |B_{(i,j)}|$. Let $s = b/p$ and $c = b/l$, where a division by zero is considered to yield zero. 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 now define the notions of TD *template* and TD *class*.

Definition 10. *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 $\mathbf{op} = \mathbf{OP} \setminus \{\bot, \top\}$. Let Θ be a nonempty 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 symbol 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 $\bot$ 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 $\bot$, 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 can now be defined.

Definition 11. *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 nonempty subset of $\mathbf{op}$. A TDMINE_Θ 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, say $\tau = X \rightarrow_\alpha Y$, where the TAR α is given by finite enumeration.

The solution to the TDMINE_Θ 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* $\text{TDMINE}_{\mathbf{op}}$.

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 $\mathcal{D} = |XY|$.*[6]

So the TDMINE_Θ 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 11 requires that the TAR α in a TDMINE_Θ problem be 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_Θ because β may not be finite. Nevertheless, this task can be readily reduced to a TDMINE_Θ problem, as follows: Let $\eta = \{(i,j) \in Future \mid j \leq \mathcal{N}\}$, 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 $\Phi \rightarrow_{\eta\cap\beta} \Psi$ belongs to the solution of the TDMINE_Θ 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.

Definition 12. *Let Θ be a nonempty subset of $\mathbf{op}$. A TDMINE(D)_Θ 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 11.*

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 $I \models^s_c \sigma$ for some $s \geq ts$ and $c \geq tc$. TDMINE(D) *is a shorthand for* $\text{TDMINE(D)}_{\mathbf{op}}$.

So, TDMINE(D) asks whether a specified support and confidence can be attained by some TD of a given TD class. Obviously, TDMINE_Θ is at least as hard as TDMINE(D)_Θ: If we have a polynomial-time algorithm for TDMINE_Θ, then we certainly do for TDMINE(D)_Θ. However, it turns out

6. Concatenation is used for the union. That is, $XY = X \cup Y$.

that TDMINE(D)_Θ is NP-complete for certain Θ, as shown in the next section.

TDMINE(D)_Θ 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 9 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.

5.3 Complexity

In this section, we explore the complexity of TDMINE(D)_Θ. This leads to the following interesting and important results:

- TDMINE(D) is **NP**-complete.
- $\text{TDMINE(D)}_{\{<,=,>\}}$, 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 4. TDMINE(D) *is* **NP**-complete.

Proof. TDMINE(D) can be solved by a *nondeterministic* 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 such that $t(x) = a$ and $t(y) = 0.5$ if $y \neq x$ $(a \in \{0, 0.5, 1\})$. 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 $j \in \{0,1,2\}$,

- $t_{ij}(x) = 0$ 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 nonnegated,
- $t_{ij}(r) = j/2$, and
- $t_{ij}(x) = 0.5$ otherwise.

	x_1	x_2	x_3	x_4	...	x_v	r	
I_1 :	.5	.5	.5	.5	...	.5	.5	
I_{21} :	0	.5	.5	.5	...	.5	.5	
	.5	0	.5	.5	...	.5	.5	
	.5	.5	0	.5	...	.5	.5	
					...			
	.5	.5	.5	.5	...	0	.5	
	1	.5	.5	.5	...	.5	.5	
	.5	1	.5	.5	...	.5	.5	
	.5	.5	1	.5	...	.5	.5	
					...			
	.5	.5	.5	.5	...	1	.5	
I_{22} :	0	0	1	.5	...	.5	0	3 tuples cor-
	0	0	1	.5	...	.5	.5	responding to
	0	0	1	.5	...	.5	1	$\neg x_1 \vee \neg x_2 \vee x_3$
				...				

Fig. 3. Construction example.

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 Fig. 3. 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 polynomial time.

Assume that for some $\Phi \rightarrow_{NextOne} \Psi$ of $[\![\tau]\!]$, we have $I \models_c^s \Phi \rightarrow_{NextOne} \Psi$ 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, say $3n$. 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, $I \models_c^s \Phi \rightarrow_{NextOne} \Psi$ 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 $\Phi(x_1) = \text{“}\leq\text{”}$ or $\Phi(x_2) = \text{“}\leq\text{”}$ or $\Phi(x_3) = \text{“}\geq\text{”}$. In general, let B be a truth assignment to the variables of V satisfying ($i \in [1..v]$):

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

Then, $n = 0$ 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 $\Phi \rightarrow_{NextOne} \Psi \in [\![\tau]\!]$, $I \models_c^s \Phi \rightarrow_{NextOne} \Psi$ with $s \geq ts$ and $c \geq tc$. To see that R can be computed in polynomial time, note that $R(\Delta)$ can be written directly from Δ. This concludes the proof. □

Remark: The TD mined in Theorem 4 only uses the operators $\leq$ and $\geq$. This leads to the following corollary.

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

Proof. This follows immediately from the fact that in the proof of Theorem 4, the TD $\Phi \rightarrow_{NextOne} \Psi$ belongs to $[\![\tau]\!]^{\{\leq,\geq\}}$. □

The following theorem states that if one allows only the operators $<$, $=$, and $>$, then the resulting TD mining problem is in **P**.

Theorem 5. $\text{TDMINE(D)}_{\{<,=,>\}}$ *is in* **P**.

Proof. Consider the $\text{TDMINE(D)}_{\{<,=,>\}}$ problem

$$(U, I, \tau, ts, tc)$$

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. □

5.4 Algorithmic Aspects

In this section, we discuss algorithms to solve certain TDMINE problems. We first give a polynomial time algorithm for $\text{TDMINE}_{\{<,=,>\}}$ and then we discuss a significant variant with fairly reduced time requirements.

5.4.1 Solving $\text{TDMINE}_{\{<,=,>\}}$

Theorem 5 suggests a naive way to solve a $\text{TDMINE}_{\{<,=,>\}}$ problem (U, I, τ, ts, tc). We now present a better approach.

Definition 13. *Let U be a set of attributes. A* comparator *over U is a DAS Φ over U such that for each $A \in U$, $\Phi(A) \in \{<, =, >\}$. 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 [13, p. 161]. To solve (U, I, τ, ts, tc) with $\tau = X \rightarrow_\alpha Y$, we proceed in three steps:

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

Fig. 4. Comparators over XY ordered by $A_1, A_2, \ldots, A_6$.

- **Step 1**. 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

$$\sum_{i=1}^{\mathcal{N}} \left(\sum_{j=i}^{\mathcal{N}} |I_i| \cdot |I_j| \right);$$

  that is, $\mathcal{O}(\mathcal{C}^2)$.
- **Step 2**. 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 Step 1 is ordered by ascending $A_1, \ldots, A_n$ in $\mathcal{O}(\mathcal{C}^2 \log \mathcal{C})$ time. We order $<$ before $=$ and $=$ before $>$. Fig. 4 shows an ordered list of comparators over XY with $X = \{A_1, A_2, A_3\}$ and $Y = \{A_4, A_5, A_6\}$.
- **Step 3**. Finally, the solution to the $\text{TDMINE}_{\{<,=,>\}}$ problem can be computed in a sequential scan of the ordered list of $\mathcal{O}(\mathcal{C}^2)$ comparators. For example, from the first seven comparators in Fig. 4, 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 $3/p$, where p is equal to $\sum_{(i,j)\in\alpha} |I_i \times I_j|$. The algorithm for this step is shown in Fig. 5.

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 $\mathcal{O}(\mathcal{C}^2)$ or more time may be very expensive in practical applications, we next derive an interesting variant with reduced time requirements.

5.4.2 Entity Evolution

TDMINE_Θ 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 disequality ($\neq$) 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

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.
% Compute 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

Fig. 5. Algorithm to compute TDs from list of comparators.

primary key K of the 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_Θ 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:

- **Step 1**. 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. If $i = j$, then every tuple of I_i is compared with itself, as no two distinct tuples of the same timeslice agree on K. If $i \neq j$, then tuples with corresponding K-values can be found in time proportional to $|I_i| + |I_j|$ by a simple merge algorithm, for we assume that I_i and I_j are ordered by K. Consequently, this step considers at most

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

tuple pairs. It can be proven by simple induction on $\mathcal{N}$ that the latter expression equals $\mathcal{NC}$. Hence, L contains at most $\mathcal{NC}$ comparators.
- **Step 2 and Step 3**. The further processing does not differ from the one in Section 5.4.1. 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.

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 Step 1 will contain not more than $2\mathcal{C}$ comparators (instead of $\mathcal{NC}$).

Example 4. The algorithm is further illustrated by Fig. 6. 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 Step 1 and the ordered list computed in Step 2. 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 $2/4$.

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	$>$	$>$

Fig. 6. Entity evolution.

To conclude, 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 $\mathbf{P} = \mathbf{NP}$. 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, for example, tuple reduction by generalization [14] or sampling [15], attribute reduction techniques [16], and incremental maintenance of mined rules [17].

6 Related Work

6.1 Database Constraints

TDs extend functional dependencies (FDs) in two ways: first, by comparing tuples over time and, second, by comparing attributes for equality as well as order ($<, >, \leq, \geq$) and disequality ($\neq$).

For nontemporal relational databases, the extension of FDs with order constraints was first explored by Ginsburg and Hull [13], [18]. Baudinet et al. [12] generalize *equality-generating dependencies*, which subsume FDs by replacing equality requirements by constraints on an interpreted domain. Interesting classes of constraint-generating dependencies are then obtained by limiting the constraints to equality, disequality, and order.

For temporal databases, the combination of FDs and time granularity was first studied by Wang et al. [4] and later extended for complex objects [8]. The notion of time granularity is captured in our framework by chronologies. To the best of our knowledge, TDs are the first dependencies that combine time and order constraints in a single construct. A more in-depth comparison of TDs with other temporal dependencies can be found in [6].

6.2 Data Mining

Recently, there has been a vast amount of interest in the discovery of different types of *association rules* from large relational tables. Association rules can take different forms [5], [19], [20]. 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 [20]) *"Married employees between 30 and 49 years old have two cars,"* which can be expressed as the universal closure of

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

and which is commonly abbreviated to

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

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. Our notions of support and confidence not only have the same name, but also the same intention.

Importantly, 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 [12], TDs are constraint-generating 2-dependencies, whereas classical association rules are constraint-generating 1-dependencies. The discovery of roll-up dependencies (RUD) proposed in [21] and the inference of functional dependencies [22] are other examples of mining constraints involving two tuples.

Most work on data mining concerns in the first place the performance of algorithms. Examples are numerous. 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.

7 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 is **NP**-hard. An upper bound for the complexity depends on the formalism used to represent TARs.

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 $\mathcal{D}$. Although solving TDMINE is generally expensive, we worked out an interesting variant with acceptable time requirements.

Acknowledgments

The author would like to thank all the anonymous referees who provided many helpful comments on a previous draft.

References

[1] J. Chomicki, "Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding," *ACM Trans. Database Systems,* vol. 20, no. 2, pp. 148–186, June 1995.

[2] J. Chomicki and D. Niwinski, "On the Feasibility of Checking Temporal Integrity Constraints," *J. Computer and System Sciences,* vol. 51, no. 3, pp. 523–535, 1995.

[3] C.S. Jensen, R.T. Snodgrass, and M.D. Soo, "Extending Existing Dependency Theory to Temporal Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 4, pp. 563–582, July/Aug. 1996.
[4] X.S. Wang, C. Bettini, A. Brodsky, and S. Jajodia, "Logical Design for Temporal Databases with Multiple Granularities," *ACM Trans. Database Systems,* vol. 22, no. 2, pp. 115–170, 1997.
[5] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 207–216 1993.
[6] J. Wijsen, "Reasoning About Qualitative Trends in Databases," *Information Systems,* vol. 23, no. 7, pp. 469–493, 1998.
[7] J. Wijsen and R. Meersman, "On the Complexity of Mining Temporal Trends," *Proc. 1997 ACM SIGMOD Int'l Workshop Research Issues on Data Mining and Knowledge Discovery,* R. Ng, ed., Technical Report 97-07, The Univ. of British Columbia, Dept. of Computer Science, pp. 77–84. 1997.
[8] J. Wijsen, "Temporal FDs on Complex Objects," *ACM Trans. Database Systems,* vol. 24, no. 1, pp. 127–176, 1999.
[9] V. Vianu, "Dynamic Functional Dependencies and Database Aging," *J. ACM,* vol. 34, no. 1, pp. 28–59, 1987.
[10] *Temporal Databases: Theory, Design, and Implementation.* Database Systems and Applications Series, A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass eds., Redwood, Calif.: Benjamin/Cummings, 1993.
[11] J. Clifford, A. Crocker, and A. Tuzhilin, "On Completeness of Historical Relational Query Languages," *ACM Trans. Database Systems,* vol. 19, no. 1, pp. 64–116, 1994.
[12] M. Baudinet, J. Chomicki, and P. Wolper, "Constraint-Generating Dependencies," *J. Computer and System Sciences,* vol. 59, no. 1, pp. 94–115, 1999.
[13] S. Ginsburg and R. Hull, "Order Dependency in the Relational Model," *Theoretical Computer Science,* vol. 26, pp. 149–195, 1983.
[14] J. Han, Y. Cai, and N. Cercone, "Data-Driven Discovery of Quantitative Rules in Relational Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 5, no. 1, pp. 29–40, Jan./Feb. 1993.
[15] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining Optimized Association Rules for Numeric Attributes," *Proc. ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems,* pp. 182–191, 1996.
[16] X. Hu and N. Cercone, "Mining Knowledge Rules from Databases: A Rough Set Approach," *Proc. Int'l Conf. Data Eng.,* pp. 96–105, 1996.
[17] D.W. Cheung, J. Han, V.T. Ng, and C.Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," *Proc. Int'l Conf. Data Eng.,* pp. 106–114, 1996.
[18] S. Ginsburg and R. Hull, "Sort Sets in the Relational Model," *J. ACM,* vol. 33, pp. 465–488, 1986.
[19] J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," *Proc. Int'l Conf. Very Large Data Bases,* pp. 420–431, 1995.
[20] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 1–12, 1996.
[21] J. Wijsen, R.T. Ng, and T. Calders, "Discovering Roll-Up Dependencies," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 213–222, 1999.
[22] J. Kivinen and H. Mannila, "Approximate Inference of Functional Dependencies from Relations," *Theoretical Computer Science,* vol. 149, pp. 129–149, 1995.

Jef Wijsen received the MS degree in geography, the MS degree in computer science, and the PhD degree in computer science from the Catholic University of Leuven, Belgium, in 1986, 1989, and 1995, respectively. Between 1995 and 1999, he was a postdoctoral researcher, first at the Free University of Brussels and later at the University of Antwerp. Since October 1999, he has been a faculty member at the University of Mons-Hainaut, Belgium. His research interests include temporal databases, spatio-temporal databases, and data mining. He is a member of IFIP WG2.6 (databases) and of TIMECENTER.