Differential Dependencies: Reasoning and Discovery

SHAOXU SONG and LEI CHEN, The Hong Kong University of Science and Technology

The importance of *difference* semantics (e.g., "similar" or "dissimilar") has been recently recognized for declaring dependencies among various types of data, such as numerical values or text values. We propose a novel form of *Differential Dependencies* (DDS), which specifies constraints on difference, called *differential functions*, instead of identification functions in traditional dependency notations like functional dependencies. Informally, a differential dependency states that if two tuples have distances on attributes X agreeing with a certain differential function, then their distances on attributes Y should also agree with the corresponding differential function on Y. For example, $[\mathsf{date}(\leq 7)] \rightarrow [\mathsf{price}(< 100)]$ states that the price difference of any two days within a week length should be no greater than 100 dollars. Such differential dependencies are useful in various applications, for example, violation detection, data partition, query optimization, record linkage, etc.

In this article, we first address several theoretical issues of differential dependencies, including formal definitions of DDS and differential keys, subsumption order relation of differential functions, implication of DDS, closure of a differential function, a sound and complete inference system, and minimal cover for DDS. Then, we investigate a practical problem, that is, how to discover DDS and differential keys from a given dataset. Due to the intrinsic hardness, we develop several pruning methods to improve the discovery efficiency in practice. Finally, through an extensive experimental evaluation on real datasets, we demonstrate the discovery performance and the effectiveness of DDS in several real applications.

Categories and Subject Descriptors: H.2.m [**Database Management**]: Miscellaneous

General Terms: Theory, Algorithms, Performance

Additional Key Words and Phrases: Data dependencies, differential dependencies

ACM Reference Format:
Song, S. and Chen, L. 2011. Differential dependencies: Reasoning and discovery. ACM Trans. Datab. Syst. 36, 3, Article 16 (August 2011), 41 pages.
DOI = 10.1145/2000824.2000826 http://doi.acm.org/10.1145/2000824.2000826

1. INTRODUCTION

Data dependencies, such as Functional Dependencies (FDS), are traditionally used for schema design, integrity constraints, query optimization and so on, with respect to schema quality in databases. Recently, data dependencies have been revisited for the quality of data, such as capturing data inconsistency [Bohannon et al. 2007], repairing inconsistent data [Cong et al. 2007], removing data duplicates [Fan et al. 2009b], etc. Conventional dependencies, originally proposed for schema-oriented issues, are defined based on equality function, that is, attribute values are compared according to

Funding for this work was provided by RGC NSFC JOINT Grant under Project No. N_HKUST61_2/09, HKUST RPC10EG13, and NSFC Grant No. 60736013, 60803105, 60873022, and 60903053.
Authors' addresses: S. Song and L. Chen, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong; email: {sshaoxu, leichen}@cse.ust.hk.

DOI 10.1145/2000824.2000826 http://doi.acm.org/10.1145/2000824.2000826

equality. However, in the data-oriented practice, besides equality, *difference* semantics (e.g., "similar" or "dissimilar") are also very useful for declaring dependencies among various types of data, such as numerical values or text values (see the following examples). Therefore, in this study, we propose a novel type of dependencies, called *Differential Dependencies* (DDs), which specify constraints on differences.

Given a relation R, a Differential Dependency (DD) has a form $\phi_L[X] \rightarrow \phi_R[Y]$, where $\phi_L[X]$ and $\phi_R[Y]$ are differential functions specifying constraints on distances over attributes X and Y of R, respectively. It states that for any two tuples from R, if their value differences (measured by a certain distance metric) on attributes X agree the differential function $\phi_L[X]$ (i.e., distance constraints on X), then their value differences on Y should also agree with the differential function $\phi_R[Y]$. As illustrated in the following examples, DDs are applicable in a broad class of data, such as categorized data, text/strings, numerical values, etc.

1.1. Motivation Examples

A DD in a credit card transaction database can be

$$\text{DD}_1 \quad [\mathsf{cardno}(=0) \wedge \mathsf{position}(\geq 60)] \rightarrow [\mathsf{transtime}(\geq 20)],$$

where $\mathsf{cardno}(=0)$ states a constraint that two transactions have the same credit card number (the distance on attribute cardno is 0), and $\mathsf{position}(\geq 60)$, $\mathsf{transtime}(\geq 20)$ are differential functions specified on position, transtime attributes. It states that if the distance of two transaction positions of a same cardno is greater than 60 km (e.g., two different cities), they are probably two transactions happening at different times, that is, the difference between transtime should be larger than 20 mins. If two card transactions do not satisfy this DD_1, that is, agreeing with the left-hand side function of DD_1 but not the right-hand side, one of the transactions could be a fraud.

Given another example in decision support systems, a DD in a price database of a flight

$$\text{DD}_2 \quad [\mathsf{date}(\leq 7)] \rightarrow [\mathsf{price}(\leq 100)]$$

states that the price difference of any two days in a week length should be less than 100 \$. Instead of a week length, another DD may specify the price difference constraint of two days not within a week but within a month.

$$\text{DD}_3 \quad [\mathsf{date}(> 7, \leq 30)] \rightarrow [\mathsf{price}(\geq 100, \leq 900)]$$

Both DD_2 and DD_3 specify constraints on the same set of attributes $\mathsf{date} \rightarrow \mathsf{price}$ but with different semantics, that is, week and month.

As a special case, if all the differential constraints are $(=0)$, DDs express the equality semantics, that is, DDs could subsume FDs. Two recent proposals, Matching Dependencies (MDs) [Fan et al. 2009b] and Metric Functional Dependencies (MFDs) [Koudas et al. 2009], also address the semantics of "similar." For example, MDs incorporate "similar" semantics in the left-hand side for record matching purposes, while MFDs use similarity metrics in the right-hand side which can be utilized in detecting violations. Compared with conventional dependencies as well as MDs and MFDs, our DDs address more general difference constraints with various semantics such as "similar" (e.g., DD_2), "dissimilar" (e.g., DD_1), or even more complicated ones (e.g., DD_3). In addition, DDs allow setting difference constraints on both determinant and dependent attributes, which makes DDs applicable to more generic and fuzzy datasets. We will use several application examples to demonstrate the wide application range of DDs.

We also note that constraints have been proposed in the literature to replace the equality semantics, by involving orderings on attribute domains. For example, Order Dependencies (ODs) proposed by Ginsburg and Hull [1983a, 1986] generalize FDs by comparing attributes for equality ($=$) as well as order ($<, >, \leq, \geq$), while Ng [1999, 2001] proposes Ordered Functional Dependencies (OFDS) to address more advanced semantics of domain orderings, that is, pointwise orderings and lexicographical orderings. Instead of order constraints between tuples, in this study, we focus on the constraints of distances between tuples.

1.2. Applications

In the following, we briefly describe several DDS application scenarios. Please refer to Section 7 for more details. (Specifically, the proposed DDS are compared with conventional dependencies in these applications, and the superiority of our proposal is verified by real dataset evaluation.)

Integrity Constraints. Similar to traditional FDS, DDS can also be utilized to detect violations or inconsistencies. For instance, according to DD_1, those tuples of a same cardno in a transaction database, which have position distance ≥ 60 km but transtime distance < 20 mins, are detected as violations to the constraint. Previous dependencies with semantics of "similar" or equality (such as FDS, MDS, or MFDS) cannot address such violations.

Query Optimization. Semantic query optimization [Chakravarthy et al. 1990; Levy and Sagiv 1995] utilizes integrity constraints (e.g., FDS) to optimize the evaluation of queries. We can also utilize DDS to rewrite queries, for example, on text/numerical attributes, where traditional FDS based on equality function are not applicable. For instance, suppose that a query selects tuples in a price database whose date distance is ≤ 7 and price difference is ≤ 100. According to DD_2, we can equivalently rewrite the query by only using one constraint, that is, date(≤ 7).

Data Partition. A typical data partition operator (e.g., group-by) divides a large relation instance into manageable partitions, where tuples in a partition satisfy certain constraints. For example, a partition scheme may require that the distances on all attributes of tuples in a partition should be less than 5. Suppose that a DD indicates that if the distance on some attributes is less than 5, that is, a *differential key*, then all the other attributes must have distance less than 5 as well. Therefore, we can equivalently use the smaller set of attributes to partition the data, which is obviously more efficient.

Record Linkage. Dependencies can be utilized in record linkage [Chaudhuri et al. 2007; Fan et al. 2009b]. For example, $[\text{name}(\leq 7) \wedge \text{address}(\leq 2)] \rightarrow [\text{ssn}(\rightleftharpoons)]$ represents a typical matching rule and can be used in a rule-based record linkage method [Hernández and Stolfo 1995]. It identifies those duplicate tuples which denote the same entity (i.e., identified $\rightleftharpoons$ on Y) according to certain $\phi_L[X]$. Different from Matching Dependencies (MDS) [Fan et al. 2009b] with only one distance constraint on each attribute, we can have various differential functions (distance constraints) for one attribute. Therefore, more than one reasonable matching rules can be declared on the same attributes (see Section 7 for details).

1.3. Contributions

In this article, we address a series of fundamental and practical issues for differential dependencies. Our main contributions in this work are listed as follows.

Foundation. We propose a novel form of *differential dependencies* and *differential keys*. To our best knowledge, this is the first work on addressing differential dependencies between attributes upon difference semantics, with both theoretical and practical consideration. Formal definitions, syntax, and properties for differential dependencies are carefully presented.

Consistency. We study the consistency problem of DDs. Given a set Σ of DDs over relation R, it is to decide whether there exists a nonempty instance I of relation R, such that Σ holds in I, written by $I \vDash \Sigma$. As we proved, the consistency problem in general is NP-complete. Thus, we identify several special cases. In the nondisjoint complement case, where the intersection of complements of differential functions are not infeasible, Σ is always consistent. In the unary LHS case, where the left-hand side differential function of each DD in Σ specifies a constraint on only one attribute, the consistency problem becomes tractable. Finally, we study the problem of finding maximum subset of DDs that are consistent.

Implication. We present a careful study of implication for DDs. It starts from a subsumption order relation of differential functions, which raises the logical implication of DDs. Given a set Σ of DDs, to imply a DD is essentially to compute the corresponding closure with respect to Σ. An inference system is then presented and proved sound and complete. Due to the implication, there may exist redundancy in an arbitrary set of DDs, that is, some DDs can imply others. Consequently, it leads us to find a concise set of DDs, that is, minimal cover for a DDs set.

Validation. We study the validation problem of DDs in a relation instance. Given a DD $\phi_L[X] \rightarrow \phi_R[Y]$ and an instance I over relation R, it is to determine whether $\phi_L[X] \rightarrow \phi_R[Y]$ holds in I, that is, $I \vDash \phi_L[X] \rightarrow \phi_R[Y]$. Unlike checking the traditional functional dependencies with equality function in $O(|I| \log |I|)$ time [Kivinen and Mannila 1995], by sorting I with respect to X values, the validation problem for DDs processes in $O(|I|^2)$ time for a predefined R. Thereby, we identify a special case where metrics in 1D space is adopted for each attribute. The validation cost then becomes $O(|I| \log^m |I|)$, where m is the number of attributes in a DD in standard form.

Discovery. We study the discovery of DDs and differential keys from a given sample data. Unfortunately, even a minimal cover of discovered DDs can be exponentially large in size with respect to the number of attributes of relation schema. Recognizing such intrinsic hardness, thereby, we focus on pruning techniques by utilizing the unique properties such as the subsumption order relation, which do not appear in previous dependency discovery scenarios. We conduct an extensive experimental evaluation on real datasets. The performance evaluation illustrates the efficiency of advanced discovery algorithms, which achieve several orders of magnitude improvement compared with a brute-force one.

Application. Finally, we illustrate the details for several applications of differential dependencies and differential keys, including the application details of violation detection, data partition, and record linkage. The effectiveness evaluation on real data demonstrates the superiority and usefulness of our DDs in various applications.

1.4. Organization

The remainder of this article is organized as follows. First, we introduce some related work in Section 2. Section 3 introduces the preliminaries of definition and syntax. We study the consistency problem in Section 4 and present two special cases of nondisjoint complement and unary LHS. Section 5 reports the implication analysis, which raises the problem of finding minimal cover. Section 6 presents the discovery of differential dependencies from data, as well as the exact validation of DDs. The details and real data evaluation of several application of DDs are also studied in Section 7. Finally, we conclude this article and outline the future work in Section 8. Table I lists the frequently used notations in this work.

Table I. Notations

Symbol	Description
R	relation schema, with $\|R\|$ attributes
I	instance of relation R, with $\|I\|$ tuples
$\phi[X]$	differential function
$\phi_1[X] \wedge \phi_2[Y]$	intersection of differential functions
$\phi_1[X] \succeq \phi_2[X]$	subsumption of differential functions
$\overline{\phi[X]}$	complement of differential function
$\phi_L[X] \rightarrow \phi_R[Y]$	differential dependency
Σ	differential dependencies set, DDs set
Σ_c	minimal cover for a DDs set

2. RELATED WORK

Conventional dependencies, originally proposed for schema issues, are often defined based on equality functions, that is, attribute values are compared with respect to equality. However, in the data-oriented practice, besides equality, difference semantics (e.g., similar or dissimilar) are also very useful for declaring dependencies among various types of data, such as numerical values or text values. In the following, we briefly review (recent) work on data dependencies under various types of data.

2.1. Identical Data

Recently, data dependencies traditionally used for schema design [Abiteboul et al. 1995] are revisited and extended for new applications like improving the quality of data. For example, Conditional Functional Dependencies (CFDs) are first proposed in Bohannon et al. [2007] for data cleaning. Cong et al. [2007] study the detecting and repairing methods of violation by CFDs. Fan et al. [2008b] investigate the propagation of CFDs for data integration. Bravo et al. [2008] propose an extension of CFDs by employing disjunction and negation. Golab et al. [2008] define a range tableau for CFDs, where each value is a range. In addition, Bravo et al. [2007] propose Conditional Inclusion Dependencies (CINDs), which are useful not only in data cleaning, but also in contextual schema matching. Ilyas et al. [2004] study a novel soft FD, where the value of X determines the value of Y not with certainty, but merely with high probability. Instead of considering difference semantics as studied in our work, all these extensions are based on the equality function. We study several typical works as follows.

Conditional Functional Dependencies. The Conditional Functional Dependencies (CFDs), as an extension of traditional FDs with conditions, are first proposed in Bohannon et al. [2007] for data cleaning. The basic idea of CFDs is making the FDs, originally holding for the whole table, valid only for a set of tuples specified by the conditions. A *Conditional Functional Dependency* (CFD) φ over R is a pair $(X \rightarrow A, t_p)$ where: (i) X is a set of attributes in R and A is a single attribute in R; (ii) $X \rightarrow A$ is a standard FD, embedded in φ; and (iii) t_p is a pattern tuple with attributes in X and A, where for each $B \in X \cup \{A\}$, $t_p[B]$ is either a constant "a" in $\mathsf{dom}(B)$, or an unnamed variable "_" that draws values from $\mathsf{dom}(B)$.

Successful studies are developed based on the notation of functional dependencies with conditions. Cong et al. [2007] study the detecting and repairing methods of violation by CFDs. Two strategies are investigated to improve the consistency of the data: (i) directly computing a repair that satisfies a given set of CFDs, (ii) incrementally finding a repair with updates to a clean database. Due to the hardness of repair problems, heuristic algorithms are developed as well. Fan et al. [2008b] investigate the propagation problem of CFDs. Given a set of CFDs on a data source, it is to determine whether or not such CFDs are still valid on the views (mapping) of the given data source. Such propagation is useful for data integration, data exchange, and data cleaning. Several algorithms are developed for computing the cover of CFDs that are propagated from original source to the views.

Further extensions on conditional functional dependencies have been drawn as well. Bravo et al. [2008] propose an extension of CFDs by employing disjunction and inequality, known as *extended Conditional Functional Dependencies* (eCFDs). Golab et al. [2008] define a range tableau for CFDs, where each value is a range.

Soft Functional Dependencies. Ilyas et al. [2004] study a novel soft FD, where the values of an attribute are well predicted by the values of another attribute. It is also a generalization of the classical notion of a hard FD where the value of X completely determines the value of Y. In a soft FD, the value of X determines the value of Y not with certainty, but merely with high probability. For example, in a database of cars, a soft dependency could be: model $\rightarrow$ make. Given that model $=$ 323, we know that make $=$ Mazda with high probability, but there is also a small chance that make $=$ BMW. Such soft FDs are useful in improving selectivity estimation during query optimization [Ilyas et al. 2004] and recommending secondary indices [Kimura et al. 2009].

Conditional Inclusion Dependencies. In addition, Bravo et al. [2007] propose Conditional Inclusion Dependency (CINDs), which are useful not only in data cleaning, but are also in contextual schema matching. For example, a CIND in a bank database could be: accountB(name, type $=$ 'saving') $\subseteq$ saving(name, branch $=$ 'B'). It states that for each tuple t_1 in the accountB relation (of branch B), if the type of t_1 is saving, then there must exist a tuple t_2 in saving such that t_1[name] $=$ t_2[name] and t_2[branch] $=$ B. In other words, instead of the entire account relation, this constraint is an inclusion dependency that holds only on the subset of account tuples such that type $=$ 'saving'.

2.2. Fuzzy Data

In recent work, the importance of introducing similarity metrics in dependencies has been commonly recognized. Attempts on replacing the equality function of conventional dependencies have also been made. Matching Dependencies (MDs) are first proposed in Fan [2008] for specifying matching rules for record matching. Rather than equality, MDs accept X values with distance less than certain thresholds and assert the matching of Y. A reasoning mechanism for deducing MDs from a set of given MDs is studied in Fan et al. [2009b]. Moreover, Koudas et al. [2009] also study the dependencies with distance metrics on attributes Y when given the equality on X. Korn et al. [2003] study Probabilistic Approximate Constraints (PACs) for network traffic databases. Specifically, PACs introduce tolerance and confidence parameters into integrity constraints. An aggregate f over a set of values in attributes X can specify constraints on differences between values within a single tuple. All these proposals consider the semantics of "similar," while our DDs address a more general case having arbitrary differential functions on both sides X and Y with "similar," "dissimilar," or even more complicated difference semantics.

Metric Functional Dependencies. Koudas et al. [2009] study the dependencies with similarity metrics on attributes Y when given the *exactly* matched values on X. A *Metric Functional Dependency* (MFD) over R has the form $X \rightarrow {}^{\delta}A$ where X is a set of attributes in R and A is a single attribute in R, and $\delta \geq 0$ is a threshold of metric distance on attribute A. We say that a relation instance r over schema R satisfies the MFD, if any two tuples $t_1, t_2 \in r$ having $t_1[X] = t_2[X]$ must have distance $\leq \delta$ on attribute A.

Matching Dependencies. Matching Dependencies (MDs) are first proposed in Fan [2008] for specifying matching rules for the object identification (see Elmagarmid et al.

[2007] for a survey). Consider a relation schema R, where each attribute A has a set of domain instances, denoted by $\mathsf{dom}(A)$.

A *similarity operator* $\approx$ on an attribute A is defined on the domain instances of A, $\approx$: $\mathsf{dom}(A) \times \mathsf{dom}(A) \rightarrow \{\mathsf{true}, \mathsf{false}\}$, which satisfies *reflexivity*, that is, $a \approx a$, and *symmetry*, that is, if $a \approx b$ then $b \approx a$, where $a, b \in \mathsf{dom}(A)$. It indicates true if two values are *similar*. This operator can be domain-specific or any similarity metrics, such as edit distance and cosine similarity, with a predefined threshold. For a set X of attributes, the similarity operator $\approx$ indicates true, *iff* the similarity operators on all $A \in X$ indicate true. A *matching operator* $\rightleftharpoons$ on an attribute A is also defined on the instances of A. It indicates true if two values are identical.

A *Matching Dependency* (MD) has a form $[X \approx] \rightarrow [Y \rightleftharpoons]$, where $X \subseteq R, Y \subseteq R$, and $\approx, \rightleftharpoons$ denotes the corresponding similarity/matching operators on attributes of X and Y, respectively. It states that for any two tuples from an instance of relation R, if they are similar on attributes X, then their Y values should be identical.

A reasoning mechanism for deducing MDs from a set of given MDs is studied in Fan et al. [2009b]. The MDs and their reasoning techniques can improve both the quality and efficiency of various record matching methods.

2.3. Ordered Data

Datasets with an ordered domain, such as numerical values, are prevalent, for example, time stamps, sequence numbers, sales, temperature, stock prices, and so on. It is also promising to address data dependencies on such ordered data. In the following, we introduce several typical works on such ordered data, where each attribute A has a partial ordering $\leq_A$ on $\mathsf{dom}(A)$. Note that, instead of order constraints between tuples, in this study we focus on the constraints of distances between tuples.

Order Dependencies. Order Dependencies have been widely studied [Dong and Hull 1982; Ginsburg and Hull 1983a, 1983b, 1986]. Let A be an attribute. Then the associated domain $\mathsf{dom}(A)$ is ordered by a fixed partial ordering $\leq_A$. Order dependencies consider this order to be either total or empty. Specifically, an attribute A has *total order* if, for each a, a' in $\mathsf{dom}(A)$, either $a \leq_A a'$ or $a' \leq_A a$. For each attribute A, the marked attributes of A are the formal symbols A, $A^<$, $A^\leq$, $A^>$, $A^\geq$, and so on. For any two tuples t_1, t_2, we write $t_1[A \leq]t_2$ if $t_1(A) \leq t_2(A)$.

An *Order Dependency* (OD) over R is an expression of the form $M \rightarrow N$, where M and N are marked attributes. We say that a relation instance r over schema R satisfies the OD, if any two tuples $t_1, t_2 \in r$, $t_1[M]t_2$ implies $t_1[N]t_2$.

Moreover, Wijsen [1998, 2001] extends order dependencies with a time dimension for temporal databases. Let $I = \{I_1, I_2, I_3, \dots\}$ be a temporal relation, which can be viewed as a time series of conventional "snapshot" relations, all over the same set of attributes. A *Trend Dependency* (TD) allows attributes with linearly ordered domains to be compared over time by using any operator of $\{<, =, >, \leq, \geq, \neq\}$. For example, a TD may state that $(\mathsf{SSN}, =) \rightarrow_{Next} (\mathsf{Sal}, \leq)$, where *Next* means that the constraint is specified over (I_i, I_{i+1}) in I. For each time point i, it requires comparing employee records at time i with records at the next time $i + 1$, such that salaries of employees should never decrease.

Ordered Functional Dependencies. Instead of the total or empty order considered in order dependencies, Ng [1999, 2001] considers *Ordered Functional Dependencies* (OFDs) with the semantics of two advanced domain orderings, that is, pointwise orderings and lexicographical orderings. Intuitively, pointwise orderings require each component of a data value to be greater than its predecessors and lexicographical orderings resemble the way in which words are arranged in a dictionary. For example, a tuple t_1 is less than another tuple t_2 according to a pointwise ordering $\leq^P$, if, for all $1 \leq i \leq |R|$, $t_1[A_i] \leq t_2[A_j]$. The tuple t_1 is less than another tuple t_2 according to a lexicographical ordering $\leq^L$, if there is an index $j \geq 1$ such that $t_1[A_j] < t_2[A_j]$ and, for each $i < j$, $t_1[A_i] = t_2[A_i]$.

Sequential Dependencies. Golab et al. [2009] propose sequential dependencies, which generalize ODs and can express other interesting relationships between ordered attributes. A *Sequential Dependency* (SD) is in the form of $X \rightarrow_g Y$, where $X \subseteq R$ are ordered attributes, $Y \subseteq R$ can be measured by certain distance metrics, and g is an interval. It states that when tuples are sorted on X, the distance between the Y-values of any two consecutive tuples are within interval g.

In order to make SDs valid in a subset tuples, SDs with conditions are studied as well. A *Conditional Sequential Dependency* (CSD) is a pair $(X \rightarrow_g Y, t_r)$, where $X \rightarrow_g Y$ is an embedded SD, and t_r is a range pattern tuple. Each range pattern t_r specifies a range of values of X that identify a subset of tuples over R (subsequence on X).

3. FOUNDATIONS

In this section, we formalize the definitions and properties of Differential Dependencies (DDS) in Section 3.1, and introduce a series of reasoning problems of DDS in Section 3.2.

3.1. Definitions and Properties

3.1.1. Differential Function. Instead of the equality function in traditional dependencies such as FDS, we introduce differential function with constraint on *difference* measured by *distance metric*.

Let B be an attribute in relation R and $\mathsf{dom}(B)$ is the domain of B. For each attribute B, we consider one distance metric on the domain of B, that is, $d_B : \mathsf{dom}(B) \times \mathsf{dom}(B) \rightarrow \mathbb{D}$, where $\mathbb{D}$ denotes all values of metric distances. It satisfies nonnegativity, $d_B(a, b) \geq 0$; identity of indiscernibles, $d_B(a, b) = 0$ *iff* $a = b$; symmetry, $d_B(a, b) = d_B(b, a)$; where $a, b \in \mathsf{dom}(B)$.

For example, the distance metric on a numerical attribute can be the absolute value of difference, that is, $d_B(a, b) = |a-b|$. For a text attribute, the distance metric can adopt various similarity/distance operators (see Elmagarmid et al. [2007] for a survey), for example, *edit distance* [Navarro 2001] .

Definition 3.1. A *differential function* $\phi[B]$ on attribute B specifies a constraint of *difference* over B, that is, a constraint on distance values $\mathbb{D}$ measured by distance metric d_B over B. For any tuple t_1, t_2 in an instance I of relation R, differential function $\phi[B]$ indicates *true*, if the difference of t_1 and t_2 on attribute B *agrees* with the constraint specified by $\phi[B]$, denoted by $(t_1, t_2) \asymp \phi[B]$.

The constraints of distance thresholds are specified by operators $\{=, <, >, \leq, \geq\}$. For example, given a differential function $\phi[\mathsf{name}] = [\mathsf{name}(\leq 6)]$, if the difference (e.g., measured by edit distance) between two tuples on the name attribute is $d_{\mathsf{name}}(t_1[\mathsf{name}], t_2[\mathsf{name}]) = 5$, which agrees ≤ 6, then we say that these two tuples agree the differential function $\phi[\mathsf{name}]$, denoted as $(t_1, t_2) \asymp [\mathsf{name}(\leq 6)]$.

Note that we consider a finite domain for each attribute B of a relation schema R. Thereby, we have a finite set $\mathbb{D}$ of distance values for each attribute B. Consequently, there may exist a finite set of differential functions that can be associated to an attribute B with respect to distance values $\mathbb{D}$.

A *differential function* $\phi[Z]$ on a set of attributes $Z \subseteq R$ is a set of metric distance constraints on all attributes $B_i \in Z$, respectively.

$$\phi[Z] = \bigwedge_{B_i \in Z} \phi[B_i]$$

For any two tuples t_1 and t_2 in I, the differential function $\phi[Z]$ indicates *true*, if the metric distances on all attributes Z agree the corresponding distance constraints specified by $\phi[Z]$, denoted by $(t_1, t_2) \asymp \phi[Z]$. For example, a differential function on name and age attributes, $[\text{name}(\leq 6) \wedge \text{age}(\geq 7)]$, indicates true, if two tuples have name distance ≤ 6 and age distance ≥ 7.

The *cardinality* of a differential function $\phi[Z]$ is the number of attributes specified in $\phi[Z]$. Differential constraints are unlimited on the other attributes $R \setminus Z$ unspecified in $\phi[Z]$. That is, any pair of tuples can always agree on this unlimited differential constraint.

We denote $\phi_1[Z]$ a *projection* of $\phi_1[X]$ on attributes Z, where $Z \subseteq X$. Let $\phi_2[Z]$ be a project of another differential function $\phi_2[Y]$, $Z \subseteq Y$. Then, $\phi_1[Z] = \phi_2[Z]$ denotes that ϕ_1 has the same distance constraints as ϕ_2 on Z. Since there is a single distance metric associated to each attribute, we can study the overlapping of differential functions on the same attributes.

The *intersection* of two differential functions $\phi_1[Z]$ and $\phi_2[Z]$ on the same attributes Z is also a differential function, denoted by $\phi_3[Z] = \phi_1[Z] \wedge \phi_2[Z]$, which satisfies the following.

—For any tuple pair $(t_1, t_2) \asymp \phi_1[Z]$ and $(t_1, t_2) \asymp \phi_2[Z]$, it always agrees with $\phi_3[Z]$ as well, that is, $(t_1, t_2) \asymp \phi_3[Z]$.
—For any tuple pair $(t_1, t_2) \not\asymp \phi_1[Z]$ or $(t_1, t_2) \not\asymp \phi_2[Z]$, it should not agree with $\phi_3[Z]$ either, that is, $(t_1, t_2) \not\asymp \phi_3[Z]$.

For example, we have $[\text{name}(\leq 9)] \wedge [\text{name}(\leq 7)] = [\text{name}(\leq 7)]$. Moreover, it is natural to apply the intersection between two differential functions $\phi_1[X]$ and $\phi_2[Y]$ on different attributes X and Y as well. Let $Z = X \cap Y$. It follows that

$$\begin{aligned}\phi_1[X] \wedge \phi_2[Y] &= (\phi_1[X \setminus Z] \wedge \phi_1[Z]) \wedge (\phi_2[Z] \wedge \phi_2[Y \setminus Z]) \\ &= \phi_1[X \setminus Z] \wedge (\phi_1[Z] \wedge \phi_2[Z]) \wedge \phi_2[Y \setminus Z].\end{aligned}$$

For example, we have $[\text{name}(\leq 5) \wedge \text{address}(\leq 12)] \wedge [\text{address}(\leq 10)] = [\text{name}(\leq 5) \wedge \text{address}(\leq 10)]$.

3.1.2. Differential Dependencies. Now, with the preliminaries of distance metrics and differential functions, we are ready to define differential dependencies.

Definition 3.2. A *Differential Dependency (*DD*)* over relation R has the form

$$\phi_L[X] \to \phi_R[Y],$$

where $X \subseteq R$, $Y \subseteq R$, and $\phi_L[X]$ and $\phi_R[Y]$ are two differential functions on attributes X and Y, respectively.

It states that for any two tuples, if their differences on attributes X agree with the constraints specified by differential function $\phi_L[X]$, then their differences on attributes Y should also agree with the constraints specified by $\phi_R[Y]$.

An instance I of relation R *satisfies* a DD, denoted by $I \vDash \phi_L[X] \to \phi_R[Y]$, if any two tuples t_1 and t_2 in I having metric distances $(t_1, t_2) \asymp \phi_L[X]$ must agree $(t_1, t_2) \asymp \phi_R[Y]$. For a set Σ of DDs, I satisfies Σ, denoted by $I \vDash \Sigma$, if $I \vDash \phi_L[X] \to \phi_R[Y]$ for each $\phi_L[X] \to \phi_R[Y] \in \Sigma$.

According to the preceding definitions, we have the following properties of differential dependencies. That is, any instance I, which satisfies the former DDs, will always satisfy the latter one.

PROPOSITION 3.3. *If $\phi_L[X] \to \phi_1[Z]$ and $\phi_L[X] \to \phi_2[Y]$, then $\phi_L[X] \to \phi_1[Z] \wedge \phi_2[Y]$.*

PROPOSITION 3.4. *If $\phi_L[X] \to \phi_1[Z]$ and $\phi_1[Z] \to \phi_R[Y]$, then $\phi_L[X] \to \phi_R[Y]$.*

3.1.3. Subsumption Order Relation. Intuitively, any two values that are "identical" (with distance $= 0$) can always be interpreted as "similar" (e.g., with distance ≤ 9). In other words, the semantics of "similar" subsume equality. Thereby, we study a general order relation $\succeq$ between differential functions, namely *subsumption*.

Definition 3.5. Let $\phi_1[Z]$ and $\phi_2[Z]$ be two differential functions on attributes Z, respectively. If any tuple pair $(t_1, t_2) \asymp \phi_2[Z]$ always agree $(t_1, t_2) \asymp \phi_1[Z]$, we say that $\phi_1[Z]$ *subsumes* $\phi_2[Z]$, written $\phi_1[Z] \succeq \phi_2[Z]$.

It is notable that the subsumption order relation is independent of any particular distance metric used in each attribute. That is, no matter what domain-specific distance metric is used for an attribute, we can always investigate the subsumption order relation among differential functions on this same metric of an attribute. Essentially, subsumption tells a general order relation of constraints on distance values $\mathbb{D}$, given any distance metric.

For example, $\phi_1[\text{name}] = [\text{name}(\leq 9)]$ subsumes $\phi_2[\text{name}] = [\text{name}(\leq 7)]$, denoted by $[\text{name}(\leq 9)] \succeq [\text{name}(\leq 7)]$. That is, given any metric, a distance value of name that agrees ≤ 7 will always agree ≤ 9. Given another example, a differential function $[\text{date}(\leq 30)]$ specifying date difference in a month length always subsumes $[\text{date}(> 7, \leq 30)]$ which denotes the date difference in a month length but larger than a week length.

Based on the subsumption and intersection semantics of differential functions, we have the following propositions.

PROPOSITION 3.6. *We have $\phi_1[Z] \succeq \phi_2[Z]$,* iff *$\phi_1[B_i] \succeq \phi_2[B_i], \forall B_i \in Z$.*

PROPOSITION 3.7. *If $\phi_1[Z] \succeq \phi_2[Z]$ and $\phi_2[Z] \succeq \phi_3[Z]$, then $\phi_1[Z] \succeq \phi_3[Z]$.*

PROPOSITION 3.8. *Given two differential functions $\phi_1[X]$ and $\phi_2[Y]$. Let $\phi_3[W] = \phi_1[X] \wedge \phi_2[Y]$. Then, we have $W = X \cup Y$, $X \subseteq W$, $\phi_1[X] \succeq \phi_3[X]$, $Y \subseteq W$ and $\phi_2[Y] \succeq \phi_3[Y]$.*

Here, $\phi_3[X]$ denotes a projection of $\phi_3[W]$ on attributes X, $X \subseteq W$. Similarly, $\phi_3[Y]$ is another projection of $\phi_3[W]$ on Y. For example, consider two differential functions $\phi_1[X] = [\text{name}(\leq 5) \wedge \text{address}(\leq 12)]$ and $\phi_2[Y] = [\text{address}(\leq 10)]$. Their intersection can be $\phi_3[W] = \phi_1[X] \wedge \phi_2[Y] = [\text{name}(\leq 5) \wedge \text{address}(\leq 12)] \wedge [\text{address}(\leq 10)] = [\text{name}(\leq 5) \wedge \text{address}(\leq 10)]$. Obviously, we have $\phi_1[X] \succeq \phi_3[X]$ and $\phi_2[Y] \succeq \phi_3[Y]$.

PROPOSITION 3.9. *For two differential functions $\phi_L[X]$ and $\phi_R[Y]$, if $Y \subseteq X$ and $\phi_R[Y] \succeq \phi_L[Y]$, then $\phi_L[X] \to \phi_R[Y]$.*

It states a trivial DD, for example, we always have $[\text{name}(\leq 5) \wedge \text{address}(\leq 10)] \to [\text{address}(\leq 12)]$.

3.1.4. Differential Key Dependencies. Traditionally, with equality functions we can identify all $t_1[R] = t_2[R]$ according to $t_1[K] = t_2[K]$ on a key $K \subseteq R$. Considering DDs with differential function, say $\phi_1[R]$ on all attributes of R, we want to find by what differential function (say differential key $\phi_2[K]$) we can identify all $(t_1, t_2) \asymp \phi_1[R]$, that is, $\phi_2[K] \to \phi_1[R]$. Formally, a differential key dependency is a special case of DDs defined as follows. As mentioned, differential keys are important in real applications such as data partition, query optimization, etc.

A *differential key* $\phi_2[K]$ relative to $\phi_1[R]$ is a differential function that can determine $\phi_1[R]$, that is, a *differential key dependency* $\phi_2[K] \rightarrow \phi_1[R]$ with $K \subseteq R$ and $\phi_2[K] \succeq \phi_1[K]$.

For example, a differential function [position(≥ 20)] is a differential key relative to [position(≥ 20) $\wedge$ area(≥ 5)], according to the following differential key dependency.

$$[\text{position}(\geq 20)] \rightarrow [\text{position}(\geq 20) \wedge \text{area}(\geq 5)]$$

A naïve key relative to $\phi_1[R]$ is $\phi_1[R]$ itself, that is, a differential key always exists. The cardinality of a differential key is the cardinality of the differential function.

A *Candidate Differential Key (*CDK*)* $\phi_c[K]$ is an *irreducible* differential key relative to $\phi_1[R]$, that is, there does not exist any $\phi_2[L]$ such that $L \subseteq K$, $\phi_2[L] \succeq \phi_c[L]$ and $\phi_2[L] \rightarrow \phi_1[R]$.

A CDK not only has a minimal cardinality as traditional candidate keys on FDS, but also should be the one subsuming all the other differential keys declared on the same attributes. Suppose that the aforesaid $\phi_2[L]$ exists. Since $\phi_c[K] \succeq \phi_1[K]$, we have $\phi_c[L] \succeq \phi_1[L]$. According to Proposition 3.7, it implies $\phi_2[L] \succeq \phi_1[L]$, that is, $\phi_2[L]$ is a differential key relative to $\phi_1[R]$ as well, and $\phi_c[K]$ is not irreducible.

3.2. Reasoning

Implication analysis is essential in data dependency studies [Abiteboul et al. 1995]. Although the distance metric may be specific to the domain of an attribute, as we presented, there does exist a general subsumption order relation among differential functions of an attribute, given any particular distance metric. In the following, we investigate the inference of DDS according to this general subsumption order relation, which is independent of any particular distance metric used in each attribute. To formally investigate the reasoning mechanism, let us first introduce some preliminaries.

Preliminary. The *complement* of a differential function $\phi[B]$, denoted by $\overline{\phi[B]}$, specifies constraints such that, any tuple pair not agreeing $\phi[B]$ will always agree $\overline{\phi[B]}$ and any tuple pair not agreeing $\overline{\phi[B]}$ will always agree $\phi[B]$. For example, we have $\overline{\text{price}(\leq 5)} =$ price(> 5). Given a set Σ of DDS, we say that Σ is *consistent*, if there exists a nonempty instance I of relation R such that $I \vDash \Sigma$; otherwise, inconsistent. Moreover, consider a differential function $\phi[Z]$. We say that $(\Sigma, \phi[Z])$ is consistent, if there exists a nonempty instance I such that $I \vDash \Sigma$ and there exist tuples $t_1, t_2 \in I$ such that $(t_1, t_2) \asymp \phi[Z]$; otherwise, inconsistent.

A differential function $\phi[B]$ is *unlimited*, if any instance I over relation R can always satisfy $\phi[B]$. For example, price(≥ 0) is unlimited, since any tuple pairs from R can always have price difference ≥ 0. It is also noticeable that all the attributes not specified by a $\phi[B]$ are also unlimited.

A differential function $\phi[B]$ is *infeasible*, if there does not exist a nonempty instance I over relation R, such that $I \vDash \phi[B]$. For example, price($\geq 3, \leq 2$) is infeasible, since no tuple pairs can satisfy such constraints on difference. In the following, we mainly consider DDS with differential functions that are not infeasible.

Two differential functions $\phi_1[X]$ and $\phi_2[Y]$ are *disjoint*, if their intersection is infeasible, that is, $\phi_1[X] \wedge \phi_2[Y] =$ *infeasible*. For example, price(> 10) and price(< 7) are disjoint, since there does not exist any pair of tuples that can satisfy both differential functions. Obviously, a differential function and its complement are always disjoint, that is, $\phi[X] \wedge \overline{\phi[X]} =$ infeasible.

Now, we introduce a series of reasoning problems for differential dependencies.

Consistency Problem. We study the consistency problem of DDS. Given a set Σ of DDS over relation R, it is to decide whether there exists a nonempty instance I of relation

Table II. Complexity of Problems

	differential functions		
	nondisjoint complement	unary LHS	general
consistency problem	always consistent	PTIME	NP-complete
consistent subset problem	always consistent	NP-complete	NP-hard
implication problem	PTIME	PTIME	co-NP-complete
minimal cover	PTIME	PTIME	NP-hard

R, such that Σ holds in I, written by $I \vDash \Sigma$. As we proved, the consistency problem in general is NP-complete. Thus, we identify several special cases. In the nondisjoint complement case, where the intersections of complements of differential functions are not infeasible, Σ is always consistent. In unary LHS case, where the left-hand side differential function of each DD in Σ specifies a constraint on only one attribute, the consistency problem becomes tractable. Finally, we study the problem of finding the maximum subset of DDs that are consistent.

Implication Problem. We study the implication problem of DDs. Given a set Σ of DDs and another DD $\phi_L[X] \rightarrow \phi_R[Y]$ over relation R, it is to decide whether Σ can imply this DD, written by $\Sigma \vDash \phi_L[X] \rightarrow \phi_R[Y]$. Again, the implication problem in the general case is most likely intractable. However, both unary LHS and nondisjoint complement cases are identified as tractable. We also study the problem of finding minimal cover of DDs. Given a set Σ of DDs over relation R, a minimal cover Σ_c of Σ is a minimal set of DDs which are equivalent to Σ.

Complexity. In the following, we study the complexity of problems such as consistency, implication, and computing minimal cover, which are all generally hard. First, we study the consistency problem in Section 4 and present two special cases of nondisjoint complement and unary LHS. Section 5 reports the implication analysis, which raises the problem of finding minimal cover. Table II lists the main conclusions of complexity.

4. CONSISTENCY ANALYSIS

In traditional FDs, given a set of FDs, there always exists an instance which can satisfy these FDs, that is, *consistent* [Abiteboul et al. 1995]. It is also interesting to study whether such an instance exists for DDs, where difference semantics are introduced. Obviously, not all DDs sets are consistent.

Example 4.1. We consider a set Σ with following DDs:

$$\begin{aligned} &\text{DD}_1 \quad [\text{date}(\leq 5)] \rightarrow [\text{price}(> 2, < 3)],\\ &\text{DD}_2 \quad [\text{date}(\leq 5)] \rightarrow [\text{price}(> 4, < 5)],\\ &\text{DD}_3 \quad [\text{date}(> 5)] \rightarrow [\text{price}(> 6, < 7)],\\ &\text{DD}_4 \quad [\text{date}(> 5)] \rightarrow [\text{price}(> 8, < 9)].\end{aligned}$$

It is impossible to construct a nonempty instance I such that $I \vDash \{\text{DD}_1, \text{DD}_2, \text{DD}_3, \text{DD}_4\}$. That is, the distance of any two tuples $t_1, t_2 \in I$ on price satisfy $(> 2, < 3)$ and $(> 4, < 5)$ at the same time, when they agree on [date(≤ 5)]. Similarly, it is impossible to find a nonempty instance where each tuple pair agrees [date(> 5)] and has price$(> 6, < 7)$ and $(> 8, < 9)$ at the same time. In other words, there is no relation instance that can satisfy all the preceding DDs.

4.1. Consistency Problem

Given a set Σ of DDs over relation R, the consistency problem is to decide whether there exists a nonempty instance I of relation R, such that Σ holds in I, written by $I \vDash \Sigma$. We say the set Σ of DDs *consistent* if such nonempty relation instance exists.

Recall that we say two differential functions are nondisjoint, if there exist some tuple pairs that agree both of them, that is, $\phi_1[X] \wedge \phi_2[Y] \neq$ infeasible.

As illustrated in Example 4.1, inconsistency may be introduced if the intersection of RHS differential functions of two DDs is infeasible, that is, no tuple pairs can be found which satisfy both DDs. Intuitively, to determine the consistency, we are essentially required to study the intersection of differential functions.

PROPOSITION 4.2. *For any set Σ of* DD*s, if the right-hand side differential functions of any two* DD*s in Σ are nondisjoint, then Σ is consistent.*

In other words, if $\forall \phi_3[U] \rightarrow \phi_1[X], \phi_4[V] \rightarrow \phi_2[Y] \in \Sigma$ having $\phi_1[X] \wedge \phi_2[Y] \neq$ infeasible, then Σ is consistent. That is, a nonempty instance I always exists having $I \asymp (\phi_1[X] \wedge \phi_2[Y])$ when given $\phi_3[U]$ and $\phi_4[V]$.

4.1.1. Nondisjoint Complement Case. We say that a set of differential functions $\{\phi_1[Z_1], \ldots, \phi_k[Z_k]\}$ is nondisjoint complement, if $\overline{\phi_1[Z_1]} \wedge \cdots \wedge \overline{\phi_k[Z_k]} \neq$ infeasible. In other words, there exist some relation instances I that can satisfy $I \asymp \overline{\phi_1[Z_1]} \wedge \cdots \wedge \overline{\phi_k[Z_k]}$.

Intuitively, we consider a set Σ of DDs, whose differential functions are nondisjoint complement. In other words, there always exists a nonempty instance I that does not agree on all the differential functions, including the left-hand side differential functions of any DDs in Σ. That is, such instance I satisfies Σ.

THEOREM 4.3. *For any set Σ of* DD*s, if the differential functions of* DD*s in Σ are nondisjoint complement, then Σ is always consistent.*

PROOF. Let $\{\phi_1[Z_1], \ldots, \phi_k[Z_k]\}$ be the set of all differential functions in Σ. According to nondisjoint complement, there exist some relation instances $I \asymp \overline{\phi_1[Z_1]} \wedge \cdots \wedge \overline{\phi_k[Z_k]}$. Such instances always have $I \vDash \Sigma$, that is, Σ is always consistent. $\square$

4.1.2. General Case. We now consider the general DDs with arbitrary differential functions defined on each attribute of R. Given a set Σ of DDs, there are finitely many differential functions that can be derived from Σ. Thereby, it is not surprising that the implication problem is intrinsically hard.

THEOREM 4.4. *The consistency problem of* DD*s in general case is* NP*-complete.*

PROOF. The problem is clearly in NP. Given any relation instance I, we can verify whether $I \vDash \Sigma$ in $O(|I|^2|\Sigma| \cdot |R|)$ time.

To prove the problem is NP-hard, we show a reduction from the nontautology problem which is proved NP-complete [Garey and Johnson 1979]. Given a set of variables $v_1, \ldots, v_m$, the nontautology problem is a decision problem, whose instance is a Boolean formula with the form $E = (x_1 \wedge y_1 \wedge z_1) \vee \cdots \vee (x_n \wedge y_n \wedge z_n)$, where $x_i, y_i, z_i, 1 \leq i \leq n$ are either v_j or $\overline{v_j}$, $1 \leq j \leq m$. It is to decide whether there exists an assignment of *true* and *false* values to the variables that will make the Boolean formula *false*, that is, nontautology. In the reduction, we build a relation $R = (A_1, \ldots, A_m, T)$ where each attribute A_j corresponds to the variable v_j. For each $(x_i \wedge y_i \wedge z_i)$, we define a differential function $\phi[W_i]$. If a variable v_j appears in $(x_i \wedge y_i \wedge z_i)$, then $\phi[W_i] = \phi[W_i] \wedge \phi[A_j]$; if $\overline{v_j}$ appears, then $\phi[W_i] = \phi[W_i] \wedge \overline{\phi[A_j]}$. We add two DDs into Σ for each $(x_i \wedge y_i \wedge z_i)$, that is, $\phi[W_i] \rightarrow \phi[T]$ and $\phi[W_i] \rightarrow \overline{\phi[T]}$. Note that no instance I can be found which both agrees $\phi[W_i]$ and satisfies these two DDs. This reduction can be conducted in polynomial time.

To find an assignment of all variables v_j is equivalent to find an instance I which agrees the corresponding differential functions (either $\phi[A_j]$ or $\overline{\phi[A_j]}$) on all the attributes A_j. Any instance I that can satisfy the set Σ of DDs, that is, consistency, will not agree all the $\phi[W_i]$ and consequently make all $(x_i \wedge y_i \wedge z_i)$ *false*, that is, nontautology. □

4.1.3. Unary Case. In the following, we study a special case where the implication problem becomes tractable. A set Σ of DDs is *unary* LHS, if all the DDs have differential functions specified only in one attribute in the left-hand side. Unary constraints with prime (single) attributes are not only theoretically interesting but also practically useful, for example, for decomposition in schema design [Mannila and Räihä 1989].

We now consider those DDs which may introduce inconsistency. Let A_i and A_j be prime attributes and

$$\begin{aligned}\phi[W] = \bigwedge\{\overline{\phi[A_i] \wedge \phi[A_j]} \mid \phi[A_i] \rightarrow \phi[X], \phi[A_j] \rightarrow \phi[Y] \in \Sigma,\\ \phi[X] \wedge \phi[Y] = \mathit{infeasible}\},\end{aligned} \tag{1}$$

where $\overline{\phi[A_i] \wedge \phi[A_j]}$ denotes the complement of a differential function on two attributes, having $\overline{\phi[A_i] \wedge \phi[A_j]} = \overline{\phi[A_i]} \vee \overline{\phi[A_j]}$. If there does not exist any nonempty $I \asymp \phi[W]$, that is, $\phi[W] =$ infeasible, then we cannot build any instance I such that $I \vDash \Sigma$, that is, inconsistent.

It is notable that a unary differential function specifies a difference range in a metric space. Thereby, we have the following claim.

CLAIM 4.1. *For any $\phi_1[A], \phi_2[A], \phi_3[A]$ on the same attribute A, if $\phi_1[A] \wedge \phi_2[A] \neq$ infeasible, $\phi_1[A] \wedge \phi_3[A] \neq$ infeasible and $\phi_2[A] \wedge \phi_3[A] \neq$ infeasible, then $\phi_1[A] \wedge \phi_2[A] \wedge \phi_3[A] \neq$ infeasible.*

To solve the problem of determining whether there exists $I \asymp \phi[W]$, we investigate an encoding to the 2-satisfiability problem, which can be solved in polynomial time [Aspvall et al. 1979].

THEOREM 4.5. *For any set Σ of DDs with unary LHS, the consistency problem can be solved in $O(|\Sigma|^2|R|)$ time.*

PROOF. First, each $\overline{\phi[A_i] \wedge \phi[A_j]}$ in formula 1 corresponds to a clause $(\overline{\phi[A_i]} \vee \overline{\phi[A_j]})$ in 2-satisfiability problem. The set of all clauses is in size $O(|\Sigma|^2)$ and can be found in $O(|\Sigma|^2|R|)$ time. For those differential functions having $\overline{\phi_1[A_i]} \wedge \overline{\phi_2[A_i]} \neq$ infeasible on the same attribute A_i, Claim 4.1 assures that a nonempty instance I can be built. Next, for any $\overline{\phi_1[A_i]} \wedge \overline{\phi_2[A_i]} =$ infeasible on the same attribute A_i, we add a clause $(\phi_1[A_i] \vee \phi_2[A_i])$. It ensures that no relation instance I can be built which agrees with both $\overline{\phi_1[A_i]}$ and $\overline{\phi_2[A_i]}$. In other words, the "variables" $\overline{\phi_1[A_i]}$ and $\overline{\phi_2[A_i]}$ cannot be assigned true at the same time in the 2-satisfiability assignment. The total number of clauses is $O(|\Sigma|^2|R|)$. It is known that 2-satisfiability problem is bounded in linear time. The conclusion is proved. □

4.2. Consistent Subset

Given a set Σ of DDs, the *maximum consistent subset problem* is a typical optimization problem, that is, to find a maximum number of DDs from Σ that are consistent.

THEOREM 4.6. *For any set Σ of DDs with unary LHS, the consistent subset problem is NP-complete.*

PROOF. Clearly, the problem is in NP, that is, according to Theorem 4.5, we can always verify a subset of DDs whether consistent or not with size at least k in polynomial time.

To prove the problem is NP-hard, we present a reduction from the vertex cover problem which is one of Karp's 21 NP-complete problems [Karp 1972]. Let G be a simple graph where each vertex v_i corresponds to a DD in Σ. For each pair of vertices (v_i, v_j) in G, let $\phi[A] \rightarrow \phi[X], \phi[B] \rightarrow \phi[Y]$ be the DDs corresponding to v_i, v_j respectively. We add two sets of new attributes U_{ij} and V_{ij} relative to v_i, v_j only. If there is an edge between v_i and v_j, then we append the following differential functions to corresponding DDs, $\phi[X] = \phi[X] \wedge \phi[U_{ij}], \phi[Y] = \phi[Y] \wedge \phi[V_{ij}], \phi[U_{ij}] \wedge \phi[V_{ij}] =$ infeasible and $\overline{\phi[A] \wedge \phi[B]} =$ infeasible. That is, we make these two DDs inconsistent, namely, there is no instance I that can satisfy these two DDs at the same time. On the other hand, if there is no edge between (v_i, v_j), we append differential functions such that $\phi[A_{ij}] \wedge \phi[B_{ij}] \neq$ infeasible, that is, make corresponding DDs always consistent. The reduction can be done in polynomial time.

To find a minimum vertex cover is equivalent to find a minimum set of vertices (DDs) that cover all the edges (inconsistency), while all the remaining DDs are consistent. □

Similar to the factor-2 approximation for finding minimum vertex cover, one can also repeatedly remove both DDs from Σ when $\phi[W] =$ infeasible in formula (1) to find an approximate minimum set of DDs, by removing which we can make the remaining DDs consistent.

Finally, from Theorem 4.6, we can immediately obtain the next corollary.

COROLLARY 4.7. *The consistent subset problem of* DDs *in the general case is* NP-*hard.*

5. IMPLICATION ANALYSIS

In this section, we study the classical implication analysis for differential dependencies. To give the formal specification, we first illustrate the following example on logical implication of DDs.

Example 5.1. Consider two DDs

$$\text{DD}_1 \quad [\mathsf{name}(\leq 7)] \rightarrow [\mathsf{address}(\leq 1)],$$
$$\text{DD}_2 \quad [\mathsf{address}(\leq 5)] \rightarrow [\mathsf{salary}(\leq 50)].$$

For any two tuples t_1 and t_2 having name distance ≤ 7, according to DD_1, their distance on address should be ≤ 1, that is, (t_1, t_2) agree address(≤ 5) as well. Consequently, the salary distance of t_1 and t_2 should be ≤ 50 according to DD_2. Therefore, we can imply another DD

$$\text{DD}_3 \quad [\mathsf{name}(\leq 7)] \rightarrow [\mathsf{salary}(\leq 50)].$$

Let Σ_1 and Σ_2 be two sets of DDs. We say that Σ_1 *logically implies* Σ_2, denoted by $\Sigma_1 \models \Sigma_2$, if for all relation instance I, $I \models \Sigma_1$ implies $I \models \Sigma_2$. Two sets Σ_1 and Σ_2 are *equivalent*, denoted by $\Sigma_1 \equiv \Sigma_2$, if $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

5.1. Implication Problem

Given a consistent set Σ of DDs and another DD $\phi_L[X] \rightarrow \phi_R[Y]$ over relation R, the implication problem is to decide whether Σ can imply this DD, written by $\Sigma \models \phi_L[X] \rightarrow \phi_R[Y]$.

THEOREM 5.2. *The implication problem of* DDs *in general case is co*-NP-*complete.*

PROOF. We study the complement of implication problem, that is, to determine whether exists an relation instance I such that $I \models \Sigma$ but not satisfying $\phi_L[X] \rightarrow \phi_R[Y]$, denoted by $I \not\models \phi_L[X] \rightarrow \phi_R[Y]$.

We show that the complement of implication problem is NP-complete. Note that to find an relation instance I that does not satisfy $\phi_L[X] \rightarrow \phi_R[Y]$ is equivalent to find an

I' such that $I' \vDash \phi_L[X] \to \overline{\phi_R[Y]}$. Consequently, the problem transforms to determine whether there exists a nonempty relation instance satisfying $\Sigma \cup \{\phi_L[X] \to \overline{\phi_R[Y]}\}$, which is exactly the consistency problem and proved to be NP-complete in Theorem 4.4. □

Definition 5.3 (*Closure*). The *closure* of a differential function $\phi_L[X]$ under Σ, denoted by $(\phi_L[X])^+$, is also a differential function, which computes the intersection of the set of differential functions that can be determined by $\phi_L[X]$ according to DDs in Σ.

$$(\phi_L[X])^+ = \bigwedge\{\phi_R[Y] \mid \Sigma \vDash \phi_L[X] \to \phi_R[Y]\}$$

For the DDs in Example 5.1, the closure of $[\mathsf{name}(\leq 7)]$ under $\{\text{DD}_1, \text{DD}_2\}$ is $[\mathsf{name}(\leq 7) \wedge \mathsf{address}(\leq 1) \wedge \mathsf{salary}(\leq 50)]$. According to Proposition 3.3, it is natural that $\phi_L[X] \to (\phi_L[X])^+$. The implication problem is then solved by computing the closure $(\phi_L[X])^+$.

LEMMA 5.4. *Let Σ be a set of* DDs *and $\phi_1[Z] = (\phi_L[X])^+$ be the closure of $\phi_L[X]$ with respect to Σ. Consider a* DD *$\phi_L[X] \to \phi_R[Y]$, $\Sigma \vDash \phi_L[X] \to \phi_R[Y]$* iff *$Y \subseteq Z$ and $\phi_R[Y] \succeq \phi_1[Y]$.*

PROOF. First, if $Y \subseteq Z$ and $\phi_R[Y] \succeq \phi_1[Y]$, according to Proposition 3.9, we have $\phi_1[Z] \to \phi_R[Y]$. Since $\phi_L[X] \to (\phi_L[X])^+$, the DD $\phi_L[X] \to \phi_R[Y]$ can be implied based on the transitivity in Proposition 3.4.

Moreover, if $\Sigma \vDash \phi_L[X] \to \phi_R[Y]$, then $\phi_R[Y]$ must be included in $\phi_1[Z] = (\phi_L[X])^+$ by $\wedge$ intersection referring to the definition of closure. That is, we have $\phi_1[Z] \wedge \phi_R[Y] = \phi_1[Z]$ as well. According to Proposition 3.8, it is sufficient to show $Y \subseteq Z$ and $\phi_R[Y] \succeq \phi_1[Y]$. □

In other words, to imply a DD is essentially to compute the corresponding closure of $\phi_L[X]$. Since $[\mathsf{salary}(\leq 50)]$ subsumes the projection on salary of the closure of $[\mathsf{name}(\leq 7)]$, it implies DD_3.

5.1.1. Unary Case. For the special case of unary LHS, the implication problem becomes tractable, whose complement, consistency problem, can be solved in polynomial time according to Theorem 4.5.

COROLLARY 5.5. *For any set Σ of* DDs *with unary* LHS*, the implication problem can be solved in $O(|\Sigma|^2|R|)$ time.*

5.1.2. Nondisjoint Complement Case. The implication problem also becomes tractable if the differential functions in Σ are nondisjoint complement. Intuitively, we are interested in all the RHS differential functions that can be determined by the given $\phi[X]$ with respect to Σ.

In the following, we show that the closure can be computed in polynomial time in the nondisjoint complement case. Thereby, the implication problem changes to be tractable.

THEOREM 5.6. *For any set Σ of* DDs *with nondisjoint complement differential functions, the implication problem can be solved in $O(|\Sigma| \cdot |R|)$ time.*

PROOF. Given a consistent set Σ of DDs, the closure of a differential function can be computed in $O(|\Sigma| \cdot |R|)$ time. Let $\phi_C[U]$ denote the intermedia result of closure for $\phi_L[X]$ in each step. Initially, we have $\phi_C[U] = \phi_L[X]$. In each step, for any $\phi[W] \to \phi[Z]$ such that $W \subseteq U$ and $\phi[W] \succeq \phi_C[W]$, we compute $\phi_C[U] = \phi_C[U] \wedge \phi[Z]$. The process repeats and does not terminate until no further change on $\phi_C[U]$.

Now, we prove that $\phi_C[U] = \phi[X]^+$. First, since each computing step follows Propositions 3.3 and 3.4, it is sufficient to show that $\phi_C[U] \succeq \phi[X]^+$.

Next, we present the opposite direction, that is, to prove $\phi[X]^+ \succeq \phi_C[U]$. It is to show that for any $\phi[V]$ such that $V \not\subseteq U$ or $V \subseteq U$ but $\phi[V] \not\succeq \phi_C[V]$ we have $\Sigma \not\models \phi_C[U] \rightarrow \phi[V]$. Let $\Sigma \models \phi_C[U] \wedge \phi[Z_i] \rightarrow \phi[V], 1 \leq i \leq k$, be the set of all DDs that can be implied by Σ such that $Z_i \not\subseteq U$ or $Z_i \subseteq U$ but $\phi[Z_i] \not\succeq \phi_C[Z_i]$. According to the nondisjoint complement, there must exist some relation instances $I \asymp \overline{\phi[Z_1]} \wedge \cdots \wedge \overline{\phi[Z_k]} \wedge \overline{\phi[V]}$. Since neither $\phi[Z_i]$ nor $\phi[V]$ subsumes $\phi_C[U]$, we can build a relation instance having $I \asymp \phi_C[U]$ at the same time. That is, there exists a relation instance $I \models \Sigma$ such that $I \not\models \phi_C[U] \rightarrow \phi[V]$. Consequently, $\Sigma \not\models \phi_C[U] \rightarrow \phi[V]$ is proved. □

5.2. Inference System

Recall that the subsumption order relation is generally defined, given any distance metric for an attribute. We now introduce an inference system for DDs which incorporates such a general subsumption order relation. It is notable that Propositions 3.3, 3.4, and 3.9 can be treated as inference axioms for implication analysis. Consequently, a complete and sound set of inference rules for DDs can be introduced analogous to Armstrong's axioms [Armstrong 1974].

Definition 5.7. Let Σ be a set of DDs over relation R. We define a set $\mathcal{I}$ of inference rules for DDs as follows.

(A1) If $Y \subseteq X$ and $\phi_L[Y] = \phi_R[Y]$, then $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$.
(A2) If $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$, then $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \wedge \phi_1[Z] \rightarrow \phi_R[Y] \wedge \phi_1[Z]$.
(A3) If $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_1[Z], \phi_1[Z] \preceq \phi_2[Z]$ and $\Sigma \vdash_{\mathcal{I}} \phi_2[Z] \rightarrow \phi_R[Y]$, then $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$.
(A4) If $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \wedge \phi_i[B] \rightarrow \phi_R[Y], 1 \leq i \leq k$, and $(\Sigma, \overline{\phi_1[B]} \wedge \cdots \wedge \overline{\phi_k[B]})$ is inconsistent, then $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$.

We say that a DD is *provable* from Σ using $\mathcal{I}$, denoted by $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$, if it is a member of Σ or is the result of one or more applications of the inference rules $\mathcal{I}$. For example, by using the transitivity *A3* rules of $\mathcal{I}$, we can imply DD$_3$ by using DD$_1$ and DD$_2$ in Example 5.1, denoted by $\{$DD$_1$, DD$_2\} \vdash_{\mathcal{I}}$ DD$_3$. Moreover, DDs can also be inferred by applying a sequence of inference rules.

Example 5.8. We consider a set Σ of DDs as follows.

DD$_1$ $[\mathsf{d}(\geq 1, \leq 7) \wedge \mathsf{p}(< 10)] \rightarrow [\mathsf{a}(\leq 150)]$
DD$_2$ $[\mathsf{p}(\geq 10)] \rightarrow [\mathsf{a}(\leq 100)]$

Let DD$_3$ be another DD

DD$_3$ $[\mathsf{d}(\geq 1, \leq 7)] \rightarrow [\mathsf{a}(\leq 150)]$.

We show that $\Sigma \vdash_{\mathcal{I}}$ DD$_3$ can be proved by the following steps.

1. $[\mathsf{d}(\geq 1, \leq 7) \wedge \mathsf{p}(\geq 10)] \rightarrow [\mathsf{d}(\geq 1, \leq 7) \wedge \mathsf{a}(\leq 100)]$	by *A2*
2. $[\mathsf{d}(\geq 1, \leq 7) \wedge \mathsf{a}(\leq 150)] \rightarrow [\mathsf{a}(\leq 150)]$	by *A1*
3. $[\mathsf{d}(\geq 1, \leq 7) \wedge \mathsf{p}(\geq 10)] \rightarrow [\mathsf{a}(\leq 150)]$	by *A3*
4. $[\mathsf{d}(\geq 1, \leq 7)] \rightarrow [\mathsf{a}(\leq 150)]$	by *A4*

As illustrated, the first three rules A1–A3 are analogous to Armstrong's axioms for conventional FDs, while A4 is specific for DDs. For instance, in step 1 of Example 5.8, we apply the augmentation rule A2 on DD$_2$. Step 2 uses the reflexivity rule A1. The transitivity rule A3 incorporates the subsumption order relation of differential functions, for example, we have $\mathsf{a}(\leq 100) \preceq \mathsf{a}(\leq 150)$ from step 1 and 2 to step 3. Intuitively, the result of step 3 together with DD$_1$ states that no matter what the distance value is on attribute p, it always has $[\mathsf{d}(\geq 1, \leq 7)] \rightarrow [\mathsf{a}(\leq 150)]$. Such an inference rule A4 is not considered in the traditional functional dependencies scenario. Formally, considering the result of step 3 and DD$_1$, we have $\overline{\mathsf{p}(< 10)} \wedge \overline{\mathsf{p}(\geq 10)} = \mathsf{p}(\geq 10) \wedge \mathsf{p}(< 10)$. Obviously, there are no tuples that can agree both these differential functions on attribute p. That is, $(\Sigma, \overline{\mathsf{p}(< 10)} \wedge \overline{\mathsf{p}(\geq 10)})$ is inconsistent as introduced in the preliminary, and *A4* can be applied.

THEOREM 5.9. *The set $\mathcal{I}$ of inference rules is*

—*(sound), if* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$ *then* $\Sigma \models \phi_L[X] \rightarrow \phi_R[Y]$,
—*(complete), if* $\Sigma \models \phi_L[X] \rightarrow \phi_R[Y]$ *then* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$,

for logical implication of DDs.

PROOF.
To show the soundness, we illustrate the semantics of four inference rules as follows.
A1 is modified according to the reflexivity rule in Armstrong's axioms for FDs. The soundness of A1 is self-explanatory, that is, $\phi_L[X] = \phi_L[X \setminus Y] \wedge \phi_L[Y] = \phi_L[X \setminus Y] \wedge \phi_R[Y] \rightarrow \phi_R[Y]$.
A2 is modified according to the augmentation rule in Armstrong's axioms for FDs. To illustrate the soundness of A2, we need to show that for any instance I of R, if $I \models \phi_L[X] \rightarrow \phi_R[Y]$, then $I \models \phi_L[X] \wedge \phi_1[Z] \rightarrow \phi_R[Y] \wedge \phi_1[Z]$. Specifically, let $t_1, t_2 \in I$ be any two tuples in I. If $(t_1, t_2) \asymp \phi_L[X] \wedge \phi_1[Z]$, that is, we have $(t_1, t_2) \asymp \phi_L[X]$ and $(t_1, t_2) \asymp \wedge \phi_1[Z]$ at the same time, it implies $(t_1, t_2) \asymp \phi_R[Y]$ according to $I \models \phi_L[X] \rightarrow \phi_R[Y]$. Thus, it follows $(t_1, t_2) \asymp \phi_R[Y] \wedge \phi_1[Z]$ as well and $I \models \phi_L[X] \wedge \phi_1[Z] \rightarrow \phi_R[Y] \wedge \phi_1[Z]$ is desired.
A3 extends the transitivity rule in Armstrong's axioms for FDs, by incorporating the subsumption order relation $\succeq$ of differential functions. Let $\Gamma = \{\phi_L[X] \rightarrow \phi_1[Z], \phi_2[Z] \rightarrow \phi_R[Y]\}$ such that $\phi_1[Z] \preceq \phi_2[Z]$. To illustrate the soundness of A3, we have to show that for any instance I of R, if $I \models \Gamma$, then $I \models \phi_L[X] \rightarrow \phi_R[Y]$. Let $t_1, t_2 \in I$ be any two tuples in I. If $(t_1, t_2) \asymp \phi_L[X]$, we have $(t_1, t_2) \asymp \phi_1[Z]$ according to $\phi_L[X] \rightarrow \phi_1[Z]$. It follows $(t_1, t_2) \asymp \phi_2[Z]$ since $\phi_1[Z] \preceq \phi_2[Z]$. Consequently, we can imply $(t_1, t_2) \asymp \phi_R[Y]$ based on $\phi_2[Z] \rightarrow \phi_R[Y]$ and $I \models \phi_L[X] \rightarrow \phi_R[Y]$ is proved.
A4 is different from Armstrong's axioms and specific to differential functions. Consider $\Sigma \vdash_{\mathcal{I}} \{\phi_L[X] \wedge \phi_i[B] \rightarrow \phi_R[Y] \mid i \in [1, k]\}$ such that $(\Sigma, \overline{\phi_1[B]} \wedge \cdots \wedge \overline{\phi_k[B]})$ is inconsistent. To illustrate the soundness of A4, we have to show that for any instance I of R, if $I \models \Sigma$, then $I \models \phi_L[X] \rightarrow \phi_R[Y]$. Since $(\Sigma, \overline{\phi_1[B]} \wedge \cdots \wedge \overline{\phi_k[B]})$ is inconsistent, for all relation instance $I \models \Sigma$, there do not exist any two tuples $t_1, t_2 \in I$ such that $(t_1, t_2) \asymp \overline{\phi_1[B]} \wedge \cdots \wedge \overline{\phi_k[B]}$. Or equivalently, $\forall t_1, t_2 \in I$, we have $(t_1, t_2) \asymp \phi_1[B] \vee \cdots \vee \phi_k[B]$. Let $t_1, t_2 \in I$ be any two tuples in I. There must at least exist one $\phi_j[B]$, $j \in [1, k]$, having $(t_1, t_2) \asymp \phi_j[B]$. If $(t_1, t_2) \asymp \phi_L[X]$, according to $\phi_L[X] \wedge \phi_j[B] \rightarrow \phi_R[Y]$, it follows $(t_1, t_2) \asymp \phi_R[Y]$ as well. That is, we have $I \models \phi_L[X] \rightarrow \phi_R[Y]$ for all the instance $I \models \Sigma$.
To prove the completeness, that is, $\Sigma \models \phi_L[X] \rightarrow \phi_R[Y]$ implies $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow \phi_R[Y]$, we consider two different cases, *C1* and *C2*, respectively.
Recall that by computing the closure $(\phi_L[X])^+$ of $\phi_L[X]$ under Σ, it naturally has $\phi_L[X] \rightarrow (\phi_L[X])^+$. Let $\phi_z[Z] = (\phi_L[X])^+$. According to Lemma 5.4, we have $\Sigma \models \phi_L[X] \rightarrow \phi_R[Y]$ *iff* $Y \subseteq Z$ and $\phi_R[Y] \succeq \phi_z[Y]$.
Now, we illustrate that such implication process can be conducted by using the inference system $\mathcal{I}$. First, we show that $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \rightarrow (\phi_L[X])^+$ by an induction. Let $(\phi_L[X])^+_i$ be the result of computing closure $(\phi_L[X])^+$ in the i-th step. Initially, we have $(\phi_L[X])^+_0 = \phi_L[X]$ and $\phi_L[X] \rightarrow (\phi_L[X])^+_0$ according to A1. Suppose inductively that $\phi_L[X] \rightarrow (\phi_L[X])^+_j$, $j = 0 \ldots i$, have been proved with respect to $\mathcal{I}$. Let $\phi_w[W] \rightarrow \phi_v[V] \in$

Σ be the DD selected in $(i+1)$-th step and let $\phi_z[Z_i] = (\phi_L[X])^+_i$. We consider the following possible cases.

(C1) If we have $W \subseteq Z_i$ and $\phi_w[W] \succeq \phi_z[W]$, then it computes the next $(\phi_L[X])^+_{i+1} = (\phi_L[X])^+_i \wedge \phi_v[V]$. The following steps show the proof of $\phi_L[X] \to (\phi_L[X])^+_{i+1}$.

(1.)	$\phi_w[W] \to \phi_v[V]$	in Σ
(2.)	$(\phi_L[X])^+_i \to \phi_w[W]$	by *A1, A3*
(3.)	$(\phi_L[X])^+_i \to \phi_v[V]$	by *A3*
(4.)	$(\phi_L[X])^+_i \to (\phi_L[X])^+_{i+1}$	by *A2*
(5.)	$\phi_L[X] \to (\phi_L[X])^+_{i+1}$	by *A3*

(C2) Otherwise, we do not have $W \subseteq Z_i$ and $\phi_w[W] \succeq \phi_z[W]$. Let $\phi_i[U_i] = \phi_w[U_i]$ such that $U_i \subseteq W$ and for each $B \in U_i$ either $B \notin Z_i$ or $\phi_w[B] \not\succeq \phi_z[B]$. Following similar steps in C1, we now have $(\phi_L[X])^+_i \wedge \phi_i[U_i] \to \phi_v[V]$, that is, $\phi_L[X] \wedge \phi_i[U_i] \to \phi_v[V]$ as well. We consider all the k DDs that can be computed with the form $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \wedge \phi_j[U_j] \to \phi_v[V]$, $1 \le j \le k$. There are two subcases.

(C2.1) If $(\Sigma, \overline{\phi_1[U_1]} \wedge \cdots \wedge \overline{\phi_k[U_k]})$ is consistent, there always exists a nonempty instance I of relation R such that $I \vDash \Sigma$ and there have tuples $t_1, t_2 \in I$ such that $(t_1, t_2) \asymp \overline{\phi_1[U_1] \vee \cdots \vee \phi_k[U_k]}$. Since (t_1, t_2) do not agree the left-hand side of any $\phi_L[X] \wedge \phi_j[U_j] \to \phi_v[V]$, they are not required to agree $\phi_v[V]$ either. It is sufficient to construct a relation instance with two tuples such that $\{t_1, t_2\} \vDash \Sigma$, $(t_1, t_2) \asymp \phi_L[X]$ but $(t_1, t_2) \not\asymp \phi_v[V]$. In other words, we do not have $I \vDash \phi_L[X] \to \phi_v[V]$ for all the relation instances I, that is, $\Sigma \nvDash \phi_L[X] \to \phi_v[V]$. We can safely ignore this case.

(C2.2) If $(\Sigma, \overline{\phi_1[U_1]} \wedge \cdots \wedge \overline{\phi_k[U_k]})$ is inconsistent, by applying A4, we have $\phi_L[X] \to \phi_v[V]$. Following similar steps in C1, we compute $(\phi_L[X])^+_{i+1} = (\phi_L[X])^+_i \wedge \phi_v[V]$. It also derives $\phi_L[X] \to (\phi_L[X])^+_{i+1}$.

To sum up all the cases, according to the induction, it is sufficient to prove that $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to (\phi_L[X])^+$.

Next, we show that $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$ when $\phi_z[Z] = (\phi_L[X])^+$, $Y \subseteq Z$ and $\phi_R[Y] \succeq \phi_z[Y]$ as Lemma 5.4. In particular, we have $(\phi_L[X])^+ \to \phi_R[Y]$ based on A1, A3 in $\mathcal{I}$. Since we have already shown $\phi_L[X] \to (\phi_L[X])^+$, $\phi_L[X] \to \phi_R[Y]$ is concluded by the transitivity rule A3, that is, $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$.

To conclude, the set $\mathcal{I}$ of inference rules is sound and complete for logical implication of DDs. $\square$

Besides, some other inference rules can be further extended.

(A5) *If* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$, $Z \supseteq X$ *and* $\phi_1[X] \preceq \phi_L[X]$, *then* $\Sigma \vdash_{\mathcal{I}} \phi_1[Z] \to \phi_R[Y]$.
(A6) *If* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$, $Y \supseteq Z$ *and* $\phi_R[Z] \preceq \phi_1[Z]$, *then* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_1[Z]$.
(A7) *If* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$ *and* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_1[Z]$, *then* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y] \wedge \phi_1[Z]$.
(A8) *If* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y] \wedge \phi_1[Z]$, *then* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$ *and* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_1[Z]$.
(A9) *If* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \to \phi_R[Y]$ *and* $\Sigma \vdash_{\mathcal{I}} \phi_R[Y] \wedge \phi_1[Z] \to \phi_2[W]$, *then* $\Sigma \vdash_{\mathcal{I}} \phi_L[X] \wedge \phi_1[Z] \to \phi_2[W]$.

Note that the last three rules A7–A9 are modified according to the union, decomposition, and pseudo-transitivity rules for FDs [Levene and Loizou 1999]. The semantics of A5 and A6 are based on the subsumption order relation specific to DDs, as illustrated in Figure 1. Given a DD $\phi_L[X] \to \phi_R[Y]$, according to A5, a $\phi_1[X]$ which is subsumed by the left-hand side $\phi_L[X]$ can determine the corresponding right-hand side $\phi_R[Y]$ as well. That is, $\phi_1[Z] \to \phi_R[Y]$, $Z \supseteq X$, can be implied. On the other hand, A6 states that any $\phi_1[Z]$ subsuming $\phi_R[Z]$, $Z \subseteq Y$, can also be determined by $\phi_L[X]$ according

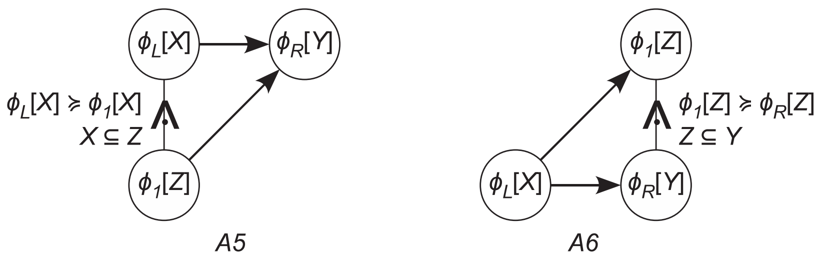

Fig. 1. Inference rules on subsumption.

to $\phi_L[X] \rightarrow \phi_R[Y]$. In fact, A5 and A6 can be proved according to A1, A2, and A3. Therefore, these secondary inference rules are derived to be sound as well.

5.2.1. Nondisjoint Complement Case. Based on the proof of C2.1 in Theorem 5.9, it is sufficient to conclude that the first three axioms in $\mathcal{I}$ are complete in the nondisjoint complement case.

COROLLARY 5.10. *The set of inference rules {A1, A2, A3} is complete and sound for logical implication of* DDs *in the nondisjoint complement case.*

5.3. Minimal Cover

According to the implication, given an arbitrary set of DDs, one may imply another which indicates the existence of redundancy. It is promising to find an equivalent concise set with less such redundancy, but without losing dependency information [Abiteboul et al. 1995].

Consider a set Σ of DDs. A *cover* Σ_1 of Σ is also a set of DDs, such that they are equivalent $\Sigma_1 \equiv \Sigma$.

Definition 5.11. A *minimal cover* Σ_c for Σ is a set of DDs such that Σ_c is logically equivalent to Σ, that is, $\Sigma_c \equiv \Sigma$, and is *minimal* according to the following properties.

(C1) (left-reduced), for any $\phi_L[X] \rightarrow \phi_R[Y] \in \Sigma_c$, there does not exist any $\phi_1[W]$ such that $W \subseteq X$, $\phi_1[W] \succeq \phi_L[W]$ and $\Sigma_c \vDash \phi_1[W] \rightarrow \phi_R[Y]$.
(C2) (right-subsumed), for any $\phi_L[X] \rightarrow \phi_R[Y] \in \Sigma_c$, there does not exist any $\phi_1[W]$ such that $Y \subseteq W$, $\phi_1[Y] \preceq \phi_R[Y]$ and $\Sigma_c \vDash \phi_L[X] \rightarrow \phi_1[W]$.
(C3) (nonredundant), there does not exist a cover Σ' of Σ such that $\Sigma' \subset \Sigma_c$.

Intuitively, these three conditions ensure that no redundancies exist in the dependencies. C3 tells that there are no redundant DDs, while C1, and C2 ensure no redundancy with respect to differential functions in left-hand side and right-hand side, respectively.

The intuition of the left-reduced property *C1* is described as follows. Suppose that such $\phi_1[W]$ exists, as illustrated in Figure 2(a). According to the reflexivity rule A1, we have $\phi_L[X] \rightarrow \phi_L[W]$. Since $\phi_1[W] \succeq \phi_L[W]$ and $\Sigma_c \vDash \phi_1[W] \rightarrow \phi_R[Y]$, we can imply $\phi_L[X] \rightarrow \phi_R[Y]$ by using the transitivity rule A3. In other words, DD $\phi_L[X] \rightarrow \phi_R[Y]$ can be logically implied from $\phi_1[W] \rightarrow \phi_R[Y]$, and thus can be removed from Σ_c as redundancy.

Similarly, the right-subsumed property *C2* is developed as follows. Again suppose such $\phi_1[W]$ exists as illustrated in Figure 2(b), having $\Sigma_c \vDash \phi_L[X] \rightarrow \phi_1[W]$, $Y \subseteq W$ and $\phi_1[Y] \preceq \phi_R[Y]$. According to the transitivity rule A3, we can directly imply $\phi_L[X] \rightarrow \phi_R[Y]$, and thus it can be removed from Σ_c as redundancy.

For example, a minimal cover for the set $\Sigma = \{\text{DD}_1, \text{DD}_2, \text{DD}_3\}$ in Example 5.1 can be $\Sigma_c = \{\text{DD}_1, \text{DD}_2\}$. That is, Σ_c can imply DD_3. By removing DD_1 or DD_2 from Σ_c, it is no longer a cover of Σ. Given another example of $\Sigma = \{\text{DD}_1, \text{DD}_2, \text{DD}_3\}$ in Example 5.8, a

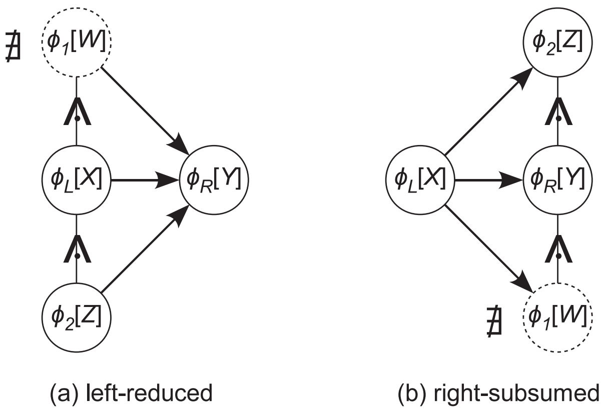

Fig. 2. Minimal cover.

minimal cover for Σ can be $\Sigma_c = \{\text{DD}_2, \text{DD}_3\}$. However, $\Sigma' = \{\text{DD}_1, \text{DD}_2\}$ is not a minimal cover, since DD_1 is not left-reduced and can be implied by DD_3 by augmentation rule *A2*.

Due to the hardness of the implication problem in the general case, it is sufficient to illustrate the hardness of finding minimal cover.

COROLLARY 5.12. *The problem of finding minimal cover for* DD*s in the general case is in the* NP*-hard.*

5.3.1. Unary Case. For the special case of unary LHS, the minimal cover can be found in polynomial time.

THEOREM 5.13. *For any set Σ of* DD*s with unary* LHS*, a minimal cover Σ_c can be found in $O(|\Sigma|^4|R|)$ time.*

PROOF. Initially, we assign $\Sigma_c = \Sigma$. In the first step of reduce, for each $\phi_L[A] \rightarrow \phi_R[Y] \in \Sigma_c$, let $\phi_1[A]$ be any differential function on the same attribute A appearing in Σ_c such that $\phi_1[A] \succeq \phi_L[A]$. If $\Sigma_c \vDash \phi_1[A] \rightarrow \phi_R[Y]$, then we replace $\phi_L[A] \rightarrow \phi_R[Y]$ by $\phi_1[A] \rightarrow \phi_R[Y]$ in Σ_c. This reduce step ensures that each DD in Σ_c is left-reduced and can be done in $O(|\Sigma|^4|R|)$ time, according to the implication cost $O(|\Sigma|^2|R|)$ in Corollary 5.5.

Next, the second step addresses the right-subsumed requirement. That is, for any pair of DDs $\phi_L[A] \rightarrow \phi_1[X], \phi_L[A] \rightarrow \phi_2[Y] \in \Sigma_c$, if $\phi_1[X] \wedge \phi_2[Y] \neq$ infeasible, we replace $\phi_L[A] \rightarrow \phi_1[X], \phi_L[A] \rightarrow \phi_2[Y]$ by $\phi_L[A] \rightarrow \phi_1[X] \wedge \phi_2[Y]$ in Σ_c.

Finally, we remove those redundant DDs from Σ_c, that is, all $\phi_L[X] \rightarrow \phi_R[Y]$ such that

$$\Sigma \setminus \{\phi_L[X] \rightarrow \phi_R[Y]\} \vDash \phi_L[X] \rightarrow \phi_R[Y].$$

This removing step can be done in $O(|\Sigma|^3|R|)$ time. □

5.3.2. Nondisjoint Complement Case. Again, if the differential functions are nondisjoint complement, we can find a minimal cover in polynomial time. Note that instead of three properties in the definition of minimal cover, it is sufficient to be minimal if a set Σ is right-subsumed and nonredundant.

LEMMA 5.14. *For any set Σ of* DD*s, if Σ is right-subsumed and nonredundant, then it must be left-reduced as well, that is, minimal.*

PROOF. According to the definition of closure, if Σ is right-subsumed, then each DD in Σ has the form $\phi_L[X] \rightarrow (\phi_L[X])^+$. Suppose that a DD is not left-reduced having

$\phi_1[Z] \rightarrow (\phi[X])^+$, $\phi_1[Z] \succeq \phi_L[Z]$, $Z \subseteq X$. Then, the set Σ is not nonredundant, that is, contradictory. □

Therefore, a minimal cover can be found by computing the closure of each left-hand side differential function in Σ and removing all the redundant DDs. Recall that Corollary 5.10 states that the inference rule set $\{A1, A2, A3\}$ is already complete and sound for logical implication of DDs in the special case of nondisjoint complement. The closure of a differential function can be computed in $O(|\Sigma| \cdot |R|)$ time and the implication cost in the nondisjoint complement case is $O(|\Sigma| \cdot |R|)$ according to Theorem 5.6.

THEOREM 5.15. *For any set Σ of* DDs *with nondisjoint complement differential functions, a minimal cover Σ_c can be found in $O(|\Sigma|^2 \cdot |R|)$ time.*

PROOF. According to Lemma 5.14, we have two steps to compute a minimal cover, the right-subsumed step with cost $O(|\Sigma|^2 \cdot |R|)$ and the nonredundant step, that is, all $\phi_L[X] \rightarrow \phi_R[Y]$ such that $\Sigma \setminus \{\phi_L[X] \rightarrow \phi_R[Y]\} \vDash \phi_L[X] \rightarrow \phi_R[Y]$, with cost $O(|\Sigma|^2 \cdot |R|)$. □

6. DISCOVERY FROM DATA

As shown in the foundations, a practical question is how to find such useful DDs, that is, the discovery problem. The discovery of data dependencies from a given relation instance has mainly two aspects of work, that is, the validation problem and discovery problem [Bitton et al. 1989; Mannila and Räihä 1992, 1994; Schlimmer 1993; Kramer and Pfahringer 1996]. The validation problem tells whether a dependency holds in a data instance, while the discovery problem returns a cover of dependencies that hold in the given data instance. In this section, given a sample relation instance I, we study the discovery of candidate differential keys and a minimal cover for all differential dependencies that hold in the given instance. Unfortunately, the discovery problem is highly nontrivial. It is known that even a minimal cover of discovered FDs can be exponentially large in size [Mannila and Räihä 1987]. Similar to CFDs discovery subsuming FDs discovery [Fan et al. 2009a], DDs discovery inherits this exponential complexity as well. In the following, we first discuss the hardness of DDs discovery and propose several pruning techniques that may improve the discovery performance in practice.

6.1. Exact Validation Problem

In this section, we study the validation problem for DDs. Given a DD $\phi_L[X] \rightarrow \phi_R[Y]$ and an instance I over relation R, the validation problem is to determine whether $\phi_L[X] \rightarrow \phi_R[Y]$ holds in I, that is, $I \vDash \phi_L[X] \rightarrow \phi_R[Y]$.

Note that to validate $\phi_L[X] \rightarrow \phi_R[Y]$ with multiple attributes in Y is essentially to validate each $\phi_R[B]$, $B \in Y$ individually. Thus, we consider the DDs in standard form $\phi_L[X] \rightarrow \phi_R[B]$ with prime attribute in the right-hand side.

A straight-forward approach is to conduct a pair-wised comparison of all tuples in I. If $\forall t_1, t_2 \in I$ having $(t_1, t_2) \vDash \phi_L[X] \rightarrow \phi_R[B]$, then we say $I \vDash \phi_L[X] \rightarrow \phi_R[B]$. Let n be the number of tuples in I and m be the number of attributes in XB. The validation can be done in $O(n^2m)$ time.

Metrics in 1D Space. Next, we investigate a special case of metrics in 1D space, for example, the metric of a numerical attribute on the absolute value of difference, that is, $d(a, b) = |a - b|$. For each $A \in XB$, the metrics in 1D space are adopted.

THEOREM 6.1. *The validation problem of* DDs *in the form of $\phi_L[X] \rightarrow \phi_R[B]$, where the metrics in 1D space are adopted for each attribute in XB, can be solved in $O(n \log^m n)$ time.*

Proof. Since metrics in 1D space are adopted in attributes XB, we can build a range tree [de Berg et al. 2008] for the tuples over XB. The validation problem is then transformed to answering range (windowing) queries as follows.

Let t be any tuple in I. The validation problem is to answer whether $\forall t' \in I$ having $(t, t') \vDash \phi_L[X] \rightarrow \phi_R[B]$, or equivalently whether not $\exists t' \in I$ such that $(t, t') \vDash \phi_L[X] \wedge \overline{\phi_R[B]}$. Recall that each differential function $\phi[A]$ specifies a range of distance on attribute A, say $A(> a_1, < a_2)$. Given a $t[A] = a$, it is to answer whether there exists a $t'[A]$ having $a - a_2 < t'[A] < a - a_1$ or $a + a_1 < t'[A] < a + a_2$, that is, $(t, t') \vDash A(> a_1, < a_2)$. We consider m attributes in XB, i.e., m-dimensional range queries. If for all $t \in I$, there does not exist a t' such that $(t, t') \vDash \phi_L[X] \wedge \overline{\phi_R[B]}$, then we say $I \vDash \phi_L[X] \rightarrow \phi_R[B]$.

It is known that a range query can be answered by range tree in $O(\log^m n + k)$ time [de Berg et al. 2008], where k is the number of tuples retrieved for a given query. We only need to find existence of t' instead of all k answers, thus it is sufficient for each query to be processed in $O(\log^m n)$ time. Since each $t[A]$ corresponds to two ranges, there are 2^m queries for each $t \in I$. The total cost of validation is $O(n \log^m n)$. □

When a predefined (fixed) relation schema R is given, that is, m is a constant, the validation problem is solved efficiently. It is notable that *fractional cascading* [Chazelle and Guibas 1986a; 1986b] can speed up the range query to $O(\log^{m-1} n + k)$ time, where k is the number of retrieved points. Consequently, a layered range tree [de Berg et al. 2008] can also be constructed for our validation problem, which yields $O(n \log^{m-1} n)$ validation time cost.

For the general case of multidimensional spaces in an attribute, for example, text attributes with cosine similarity or edit distance [Navarro 2001], the efficient validation problem is still open.

6.2. Reduction Problem

Various differential functions in attributes are the major challenge in DDs discovery, which are not considered in previous work. Given a relation instance I, for each attribute B_i, we study a finite set of differential functions defined on it, denoted by $\Phi(B_i)$. For example, the search space of differential functions based on intervals of edit distance in an attribute B_i could be $\Phi(B_i) = \{B_i(\geq u, \leq v) \mid 0 \leq u, u \leq v, v \leq D\}$, where D is the maximum edit distance value. It can also be attribute specific, such as $\Phi(\text{date}) = \{\text{date}(\leq 7), \text{date}(\leq 30), \text{date}(> 7, \leq 30), \ldots\}$, which denote constraints of date differences in a week length, in a month length, in a month length but not a week, respectively. Consequently, the search space of all m attributes in a set $X = \{B_1, \ldots, B_m\}$ is $\Phi(X) = \Phi(B_1) \times \cdots \times \Phi(B_m)$, $B_i \in X$.

According to the left-reduced condition in the definition of minimal covers and the definition of Candidate Differential Keys (CDKs), we are essentially required to find reductions as irreducible left-hand side of DDs. By solving ths reduction problem from data, we can directly discover DDs and CDKs.

Problem 6.1 (Reduction in data). *Given any relation instance I and a differential function $\phi_R[Y]$, the problem of reduction with respect to data I is to find all $\phi_L[X]$ such that:* (i) $I \vDash \phi_L[X] \rightarrow \phi_R[Y]$, (ii) *there does not exist any $\phi_1[W]$ having $W \subseteq X$, $\phi_1[W] \succeq \phi_L[W]$ and $I \vDash \phi_1[W] \rightarrow \phi_R[Y]$.*

6.2.1. The Hardness. Obviously, there may exist more than one $\phi_L[X]$ that can satisfy the preceding conditions, that is, reduction is not unique. The reduction problem is to find all such $\phi_L[X]$. Unfortunately, the size of all reductions for a $\phi_R[Y]$ in an instance I of relation R can be exponentially large in the number of attributes in R. It is already

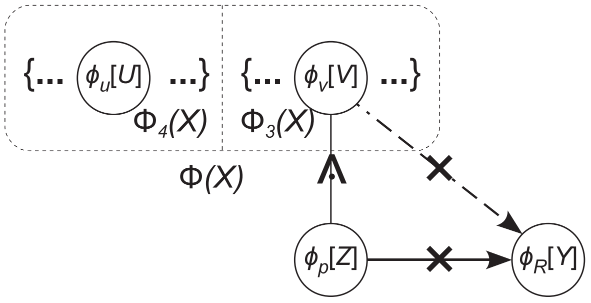

Fig. 3. Negative pivot.

known that a minimal cover of all reductions for FDs can be exponentially large in the number of attributes [Mannila and Räihä 1987]. Since FDs are considered as special cases of DDs where all the differential constraints are set to $= 0$, it is not surprising that such exponential size also carries over to DDs. Due to this intrinsic hardness, in the following, we focus on pruning techniques that may behave well in practice.

6.2.2. Brute Force. Consider a set X of m attributes. The straight-forward approach is to evaluate all possible $\phi_L[X] \to \phi_R[Y], \forall \phi_L[X] \in \Phi(X)$, in order to find a set Σ of DDs that hold in instance I. Then, we remove the DDs from Σ which are reducible and redundant.

Let s be the maximum size of the set of differential functions defined on each attribute, $s = \max |\Phi(B_i)|$. Then, the search space complexity of differential functions on m attributes is $O(s^m)$. The cost of evaluating one DD is $O(n^2)$, where n is the size of instance I. It is a costly operation especially when the instance size is large. The time complexity of brute-force approach is $O(n^2 s^m)$.

6.2.3. Negative Pruning. Instead of evaluating all possible differential functions of $\Phi(X)$ in brute-force way, we can explore the subsumption order relation for pruning in the search space $\Phi(X)$.

LEMMA 6.2. *For any $\phi_1[V], \phi_2[Z]$ having $V \subseteq Z$, $\phi_1[V] \succeq \phi_2[V]$, if $I \not\models \phi_2[Z] \to \phi_R[Y]$, then $I \not\models \phi_1[V] \to \phi_R[Y]$ either.*

For example, suppose that $\phi_2[Z] = [\text{name}(\leq 5) \wedge \text{address}(\leq 10)]$ and $\phi_2[Z] \to \phi_R[Y]$ does not hold in I. Then, we can directly infer that $\phi_1[V] \to \phi_R[Y]$, for example, $[\text{name} (\leq 7)] \to \phi_R[Y]$, should not hold in I either without evaluating it in I. A similar negative pruning technique is also developed for discovering FDs in a bottom-up search strategy [Flach and Savnik 1999]. Algorithm 1 implements this negative pruning specific to DDs in Lemma 6.2.

We develop a bottom-up search according to the negative pruning in Lemma 6.2. Specifically, let $\phi_p[Z] \in \Phi(X)$ be the current evaluation as illustrated in Figure 3. It divides the remaining $\Phi(X)$ into two subsets, that is, differential functions $\phi_v[V]$ in $\Phi_3(X)$ having $V \subseteq Z$ and $\phi_v[V] \succeq \phi_p[V]$; otherwise in $\Phi_4(X)$. If $\phi_p[Z] \to \phi_R[Y]$ does not hold in I according to the evaluation, then we can pruning all the remaining $\phi_v[V] \in \Phi_3(X)$ according to the negative pruning in Lemma 6.2. We name $\phi_p[Z]$ a *negative pivot*. Finally, the results are combined together by the $\uplus$ operator, which removes those reducible DDs.

This negative pruning approach favors certain processing order of differential functions in $\Phi(X)$. For any two differential functions $\phi_v[V], \phi_w[W]$ having $V \subseteq W$ and $\phi_v[V] \succeq \phi_w[V]$, the subsumed one, that is, $\phi_w[W] \to \phi_R[Y]$, is preferred to be evaluated first, in order to utilize potential pruning opportunity on $\phi_v[V]$. In the following, we assume that differential functions in $\Phi(X)$ have already been arranged by subsumption order. That is, for any i-th differential function $\phi_v[V] \in \Phi(X)$ and j-th differential function $\phi_w[W] \in \Phi(X)$, $i < j$, we have $V \subseteq W$ and $\phi_v[V] \succeq \phi_w[V]$. The search algorithm starts from the end of $\Phi(X)$, namely bottom-up search.

ALGORITHM 1: Bottom-Up REDUCE(I, $\Phi(X)$, $\phi_R[Y]$)

```
Input: An instance $I$, a search space $\Phi(X)$ of $\phi_L[X]$ and a target $\phi_R[Y]$
Output: A minimal set $\Sigma$ for all DDs determining $\phi_R[Y]$ under $I$
1: $\Sigma := \emptyset$
2: $\phi_p[Z] :=$ the last element removed from $\Phi(X)$
3: $\Phi_3(X) := \{\phi_v[V] \in \Phi(X) \mid V \subseteq Z, \phi_v[V] \succeq \phi_p[V]\}$
4: $\Phi_4(X) := \Phi(X) \setminus \Phi_3(X)$
5: if $I \not\models \phi_p[Z] \to \phi_R[Y]$ then
6:     $\Sigma := \Sigma \uplus$ REDUCE($I$, $\Phi_4(X)$, $\phi_R[Y]$)
7: else
8:     $\Sigma := \Sigma \uplus \{\phi_p[Z] \to \phi_R[Y]\}$
9:     $\Sigma := \Sigma \uplus$ REDUCE($I$, $\Phi_3(X)$, $\phi_R[Y]$)
10:    $\Sigma := \Sigma \uplus$ REDUCE($I$, $\Phi_4(X)$, $\phi_R[Y]$)
11: end if
12: return $\Sigma$
```

The bottom-up search procedure with negative pivot is given in Algorithm 1. This negative pruning performs well when most DDs do not hold in I. Let r be the probability that an arbitrary DD holds in I, $0 < r < 1$. The algorithm runs in $O(n^2 s^{m \log_2(1+r)})$ time. In the worst case, that is, all the candidates hold in the given instance I having $r \approx 1$, the time complexity is $O(n^2 s^m)$.

6.2.4. Positive Pruning. Recall that according to the inference rules, certain DDs may imply others for any relation instance. Intuitively, we can investigate a positive pruning based on the DDs that hold in the given relation instance I.

LEMMA 6.3. *For any $\phi_1[V], \phi_2[W]$ having $W \subseteq V, \phi_2[W] \succeq \phi_1[W]$, if $I \models \phi_2[W] \to \phi_R[Y]$, then $I \models \phi_1[V] \to \phi_R[Y]$ too.*

Suppose that a DD $\phi_2[W] \to \phi_R[Y]$ holds in I, for example, $I \models [\mathsf{name}(\leq 9)] \to \phi_R[Y]$. According to Lemma 6.3, we can prune $\phi_1[V] = \mathsf{name}(\leq 7)$ subsumed by $\phi_2[W]$, that is, $[\mathsf{name}(\leq 7)] \to \phi_R[Y]$ must hold in I as well and can be safely pruned since it is not left-reduced. Similar positive pruning techniques are also used in a top-down search strategy for discovering FDS (TANE) [Huhtala et al. 1998] and CFDS [Chiang and Miller 2008; Fan et al. 2009a]. The implementation of positive pruning specific to DDS in Lemma 6.3 is also introduced.

Intuitively, the positive pruning based on Lemma 6.3 favors the processing of candidates that subsume others. That is, we start from the beginning of $\Phi(X)$ in a top-down strategy. In each step, we consider the next irreducible differential function, say $\phi_p[W]$, as illustrated in Figure 4. It introduces another division of $\Phi(X)$, that is, $\Phi_1(X)$ records all the remaining $\phi_v[V]$ such that $W \subseteq V$ and $\phi_p[W] \succeq \phi_v[W]$; otherwise, in $\Phi_2(X)$. Once we find that $I \models \phi_p[W] \to \phi_R[Y]$, according to Lemma 6.3, all the remaining candidates subsumed by $\phi_p[W]$ in $\Phi_1(X)$ can be safely pruned. We name such $\phi_p[W]$ a *positive pivot*.

The positive pruning favors the case that most DDS hold in I, that is, the irreducible DDS can be found early and prune the remaining ones. Let r be the probability that a DD holds in I, the same as negative pruning. The top-down search algorithm with positive pivot runs in $O(n^2 s^{m \log_2(2-r)})$ time. In the worst case, that is, all the candidates

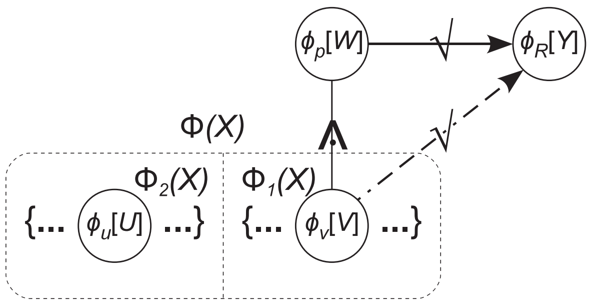

Fig. 4. Positive pivot.

do not hold in the given instance I having $r \approx 0$, the algorithm has to process the entire search space with cost $O(n^2 s^m)$. In practice, as illustrated in experiments in Section 6.4, the appearance of worst cases of positive pruning is much less than that of negative pruning. Thereby, the positive pruning approach outperforms the negative one in most cases.

6.2.5. Hybrid Pruning. It is notable that the worst case of negative pruning is that all the candidates from $\Phi(X)$ hold in the given instance, while the worst case of positive pruning is that most candidates do not hold. Nevertheless, to avoid worst cases for negative and positive pruning, we can develop a hybrid approach with both positive and negative pruning, which are used by turns. Specifically, in each step, we first evaluate candidates according to positive pruning. If the positive pruning is not valid, then the negative one is applied. As presented in experimental evaluation in Section 6.4, the hybrid approach can successfully avoid the worst cases of negative and positive pruning and keep low cost.

6.2.6. Instance Exclusion. The aforesaid approaches have to evaluate the entire I for each DD. However, according to the subsumption order relation, if one differential function subsumes another, it denotes that the set of tuples agreeing on the former one should be a superset of the latter one. Recognizing this relationship on instances, we now present an approach by avoiding evaluating the entire I.

Given a relation instance I, we define a distance instance $D(I)$ with respect to I, which considers all the pairs of tuples in I.

$$D(I) = \{(t_i, t_j) \mid \forall t_i, t_j \in I\}.$$

Given any DD $\phi_L[X] \rightarrow \phi_R[Y]$, we define $D(I, \phi_L[X], \neg\phi_R[Y]) =$

$$\{(t_i, t_j) \in D(I) \mid (t_i, t_j) \asymp \phi_L[X], (t_i, t_j) \not\asymp \phi_R[Y]\},$$

that is, the tuple pairs agreeing $\phi_L[X]$ but not agreeing $\phi_R[Y]$.

LEMMA 6.4. *An instance I satisfies a* DD*, $I \vDash \phi_L[X] \rightarrow \phi_R[Y]$,* iff $D(I, \phi_L[X], \neg\phi_R[Y]) = \emptyset$.

In other words, we want to discover a reduction $\phi_L[X]$ such that $D(I, \phi_L[X], \neg\phi_R[Y]) = \emptyset$. Now, we formalize the relationship of different DDs with respect to tuple pairs in distance instance.

LEMMA 6.5. *For any $\phi_1[V], \phi_2[W]$ having $W \subseteq V, \phi_2[W] \succeq \phi_1[W]$, we have $D(I, \phi_1[V], \neg\phi_R[Y]) \subseteq D(I, \phi_2[W], \neg\phi_R[Y])$.*

Therefore, instead of considering the entire $D(I)$, we can use $D(I, \phi_2[W], \neg\phi_R[Y])$ to compute $D(I, \phi_1[V], \neg\phi_R[Y])$. Specifically, we can always first process the one subsuming others, that is, $\phi_2[W]$. If $D(I, \phi_2[W], \neg\phi_R[Y]) \neq \emptyset$, in the following step of computing $D(I, \phi_1[V], \neg\phi_R[Y])$, we can directly utilize the distance instance with respect to $\phi_2[W]$. If $D(I, \phi_1[V], \neg\phi_R[Y]) = \emptyset$, then we return it as a reduction according to Lemma 6.4.

Similar instance exclusion techniques are also used in a depth-first search strategy for discovering FDS (FastFD) [Wyss et al. 2001] and CFDS (FastCFD) [Fan et al. 2009a]. Algorithm 2 implements the instance exclusion specific to DDS, where the subsumption order relation of differential functions is not considered in previous work.

Algorithm 2 presents the instance exclusion in the top-down search with positive pivot. Let D denote the distance instance of tuple pairs with respect to relation instance I. We call the function EXCLUDE($D, \phi_p[W] \rightarrow \phi_R[Y]$) to compute $D(I, \phi_p[W], \neg\phi_R[Y])$, say D_1, from the previous D, according to Lemma 6.5. If $D_1 = \emptyset$, then we have $I \vDash \phi_p[W] \rightarrow \phi_R[Y]$ according to Lemma 6.4.

ALGORITHM 2: Instance-exclusion REDUCE(D, $\Phi(X)$, $\phi_R[Y]$)

```
Input: A distance instance $D$, a search space $\Phi(X)$, and a target $\phi_R[Y]$
Output: A minimal set $\Sigma$ for all DDS determining $\phi_R[Y]$ under $I$
 1: $\Sigma := \emptyset$
 2: $\phi_p[W]$ := the first element removed from $\Phi(X)$
 3: $D_1$ := EXCLUDE($D, \phi_p[W] \rightarrow \phi_R[Y]$)
 4: $\Phi_1(X) := \{\phi_v[V] \in \Phi(X) \mid W \subseteq V, \phi_p[W] \succeq \phi_v[W]\}$
 5: $\Phi_2(X) := \Phi(X) \setminus \Phi_1(X)$
 6: if $D_1 \neq \emptyset$ then
 7:     $\Sigma := \Sigma \uplus$ REDUCE($D_1, \Phi_1(X), \phi_R[Y]$)
 8: else
 9:     $\Sigma := \Sigma \uplus \{\phi_p[W] \rightarrow \phi_R[Y]\}$
10: end if
11: $\Sigma := \Sigma \uplus$ REDUCE($D, \Phi_2(X), \phi_R[Y]$)
12: return $\Sigma$
```

Let l be the exclusion rate on average, $0 < l < 1$, that is, $l|D|$ tuple pairs remain after an exclusion on D in each step. Then, the algorithm runs in $O(n^2 s^{m \log_2 l(2-r)})$ time. Besides positive pruning, the instance exclusion can also cooperate with the hybrid pruning approach and keep low cost in all cases as presented in the experiments in Section 6.4.

6.3. Implementation Issues

During the implementation of the preceding REDUCE algorithms, we need to maintain the following two types of data objects.

—$\Phi(X)$, *set of differential functions*. Recall that set $\Phi(X)$ denotes the search space of the left-hand side differential functions in the reduction. To implement the positive and negative pruning, we need to support the split of set $\Phi(X)$ in two ways: (a) given a positive pivot $\phi_p[W]$, to split the set $\Phi(X)$ into $\Phi_1(X)$ and $\Phi_2(X)$ such that for each $\phi_v[V]$ in $\Phi_1(X)$, $\phi_p[W]$ subsumes the projection of $\phi_v[V]$ on W; (b) given a negative pivot $\phi_p[Z]$, to split the set $\Phi(X)$ into $\Phi_3(X)$ and $\Phi_4(X)$ such that each $\phi_v[V]$ in $\Phi_3(X)$ subsumes the projection of $\phi_p[Z]$ on V. As illustrated in Figure 4, the split on positive pivot is used in the positive pruning, which can also cooperate with the instance exclusion in line 4 in Algorithm 2. Moreover, as shown in Figure 3, the negative split is used for the negative pruning in line 3 in Algorithm 1.

—D, *set of tuple pairs with distances*. Recall that set D is a distance instance of tuple pairs in the relation instance I. To support the instance exclusion, we have to enable a filtering of tuple pairs in the set D. Specifically, given any $\phi_p[W] \rightarrow \phi_R[Y]$, the filtering removes those tuple pairs in the set D whose distance values agree $\phi_p[W]$ but do not agree $\phi_R[Y]$. The filtering can be implemented by a single pass through the elements in set D. As illustrated in Algorithm 2, different versions of distance instance D are maintained by the filtering operation EXCLUDE$(D, \phi_p[W] \rightarrow \phi_R[Y])$.

6.3.1. Discovering DDs. We are now ready to introduce the discovery of DDS embedded in a given sample data, that is, given an instance I of relation R, to find a *minimal cover* Σ_c for all DDS that hold in I.

We have studied the reduction in data for the left-reduced property. In the following, we focus on the other conditions of minimal cover, that is, right-subsumed and nonredundant. Intuitively, we can first discover a set Σ of all left-reduced DDS whose right-hand sides $\phi[Y]$ are differential functions on single attribute Y. Then, we remove redundant DDS in Σ.

ALGORITHM 3: Minimal COVER(R, I)

```
Input: A relation schema $R$, and an instance $I$ of $R$.
Output: A minimal cover $\Sigma$ for all DDS under $I$.
 1: $\Sigma := \emptyset$
 2: for each attribute $Y \in R$ do
 3:    $X := R \setminus \{Y\}$
 4:    for each $\phi_R[Y] \in \Phi(Y)$ do
 5:       $\Sigma := \Sigma \cup$ REDUCE($I$, $\Phi(X)$, $\phi_R[Y]$)          {reduce step}
 6:    end for
 7: end for
 8: repeat
 9:    if exist $\phi_L[X] \rightarrow \phi_R[Y]$, $\phi_w[W] \rightarrow \phi_v[V] \in \Sigma$ such that $X \subseteq W$, $\phi_L[X] \succeq \phi_w[X]$, $V \subseteq Y$ and
       $\phi_v[V] \succeq \phi_R[V]$ then
10:       remove $\phi_w[W] \rightarrow \phi_v[V]$ from $\Sigma$
11:    end if
12: until no changes on $\Sigma$
12: return $\Sigma$
```

Algorithm 3 presents the discovery procedure. Note that the complexity of redundant removing step (lines 6–9) depends on the number of DDS in Σ returned in the reduce step. Here, the search space of $\phi[Y]$ is in size of $s|R|$. The complexity of minimal COVER algorithm is $O(n^2|R|s^{|R|})$. The intrinsical hardness comes from the size of minimal cover, which can be exponentially large. In practice, we can apply our advanced REDUCE algorithms for discovering minimal cover. For example, as presented in Section 6.4, the instance-exclusion REDUCE can achieve several orders of magnitude improvement than the brute-force one.

6.3.2. Discovering CDKs. We study the discovery of candidate differential keys from a relation instance, that is, given an instance I of relation R and a differential function $\phi_1[R]$, to find a set of all CDKS relative to $\phi_1[R]$ under I. Recall that a CDK is a differential key that is irreducible in the key dependency, we transform the CDKS discovery to the preceding reduction problem as well, that is, reductions of $\phi_1[R]$. Specifically, for those irreducible DDS returned by the REDUCE algorithm, $\phi_L[X] \rightarrow \phi_1[R]$, whose left-hand side $\phi_L[X] \succeq \phi_1[X]$, their $\phi_L[X]$ are reported as CDKS.

6.4. Discovery Experiments

In this section, we report the experimental results on testing the efficiency of our methods in discovering DDS. In each experiment of discovery, we use an instance as input, and observe the results of CDKS and DDS that hold in the given instance. Edit

Table III. Examples of DDs in CiteSeer (10000)

DD_1	[subject(≤ 1)] $\rightarrow$ [title(≤ 2)]
DD_2	[subject($= 0$)] $\rightarrow$ [title(≤ 1)]
DD_3	[address(≤ 3) $\wedge$ subject($= 0$)] $\rightarrow$ [title($= 0$) $\wedge$ author($= 0$)]
DD_4	[affiliation(≤ 3) $\wedge$ subject($= 0$)] $\rightarrow$ [author($= 0$)]
DD_5	[affiliation(≤ 1) $\wedge$ subject(≤ 1)] $\rightarrow$ [author($= 0$)]
DD_6	[address(≤ 3) $\wedge$ affiliation(≤ 3) $\wedge$ subject(≤ 1)] $\rightarrow$ [author($= 0$)]
DD_7	[title(≤ 3) $\wedge$ author(≤ 3) $\wedge$ affiliation(≤ 3)] $\rightarrow$ [subject(≤ 3)]
DD_8	[title(≤ 3) $\wedge$ author(≤ 3) $\wedge$ address(≤ 3)] $\rightarrow$ [subject(≤ 3)]
DD_9	[title(≤ 3) $\wedge$ author(≤ 1)] $\rightarrow$ [subject(≤ 3)]
DD_{10}	[title(≤ 3) $\wedge$ author(≤ 3) $\wedge$ affiliation(≤ 1)] $\rightarrow$ [subject(≤ 2)]
DD_{11}	[title(≤ 2) $\wedge$ affiliation(≤ 2)] $\rightarrow$ [subject(≤ 2)]
DD_{12}	[title(≤ 1) $\wedge$ affiliation(≤ 3)] $\rightarrow$ [subject(≤ 2)]
DD_{13}	[title(≤ 2) $\wedge$ address(≤ 3)] $\rightarrow$ [subject(≤ 2)]
DD_{14}	[title(≤ 2) $\wedge$ author(≤ 3)] $\rightarrow$ [subject(≤ 2)]
DD_{15}	[title(≤ 1) $\wedge$ author(≤ 3) $\wedge$ affiliation(≤ 3)] $\rightarrow$ [subject(≤ 1)]
DD_{16}	[title(≤ 1) $\wedge$ author(≤ 3) $\wedge$ address(≤ 3)] $\rightarrow$ [subject(≤ 1)]
DD_{17}	[title($= 0$) $\wedge$ author(≤ 3)] $\rightarrow$ [subject(≤ 1)]
DD_{18}	[title($= 0$) $\wedge$ author(≤ 3) $\wedge$ address(≤ 3)] $\rightarrow$ [subject($= 0$)]

distance with q-grams [Gravano et al. 2001] is adopted as the distance metric. All the algorithms are implemented by Java. The experiments run on a machine with Intel Core 2 CPU (2.13 GHz) and 2GB of memory.

Real datasets are adopted in the experiments. For example, the CiteSeer[1] dataset collects records of scientific research papers including the following attributes (title, author, address, affiliation, subject, description, id). A projection of the first m attributes is conducted on the original scheme to generate the scheme with m attributes. For each attribute, we associate with 5 differential functions. It is notable that the discovered dependencies are valid with respect to the domain of the given dataset. Therefore, we also employ a secondary dataset DBLP[2] which consists of conference proceedings with schema (conference, title, booktitle, series, volume, publisher, year, id). The discovery results could be different in CiteSeer and DBLP, since they are different collections of research publications.

6.4.1. Discovery Results. In the first set of experiments, we observe the results of discovered dependencies. To evaluate the quality of discovered dependencies, we compare the results under different data sizes through a cross validation.

Table III presents all the DDs in the minimal cover discovered from a given relation instance with 10000 tuples. It is worth noting that these 18 DDs do not denote all the DDs that hold in the given instance. Instead, they indicate a minimal cover which can imply all the other DDs holding in the instance. For example, the DD_2 in Table III states that for any two tuples in the instance, if they have exactly the same subject (with distance = 0), then these two tuples have similar title (with distance ≤ 1). Since the discovered results are a minimal cover of all dependencies, any other DDs implied by Table III also hold in the instance. For instance, [subject($= 0$)] $\rightarrow$ [title(≤ 2)], which can be implied by DD_2, will also be valid. Moreover, according to the reduction, it also implies that the DDs are left-reduced. That is, there does not exist any other $\phi_L[X]$ such as [subject(≤ 1)] $\succeq$ [subject($= 0$)] that can also imply [title(≤ 1)].

Note that when more tuples are available in the given instance, the discovered results may change. For example, Table IV presents the minimal cover discovered from 20000 tuples. Note that many authors may share similar names in CiteSeer, and the ability of disambiguation by the author attribute might not be significant in large data

[1]http://citeseer.ist.psu.edu/
[2]http://www.informatik.uni-trier.de/~ley/db/

Table IV. Examples of DDs in CiteSeer (20000)

DD_1	[address(≤ 3) $\wedge$ subject(≤ 1)] $\rightarrow$ [title(≤ 3)]
DD_2	[subject($= 0$)] $\rightarrow$ [title(≤ 1)]
DD_3	[address(≤ 3) $\wedge$ subject($= 0$)] $\rightarrow$ [title($= 0$) $\wedge$ author($= 0$)]
DD_4	[affiliation(≤ 3) $\wedge$ subject($= 0$)] $\rightarrow$ [author($= 0$)]
DD_5	[title(≤ 2) $\wedge$ address(≤ 3) $\wedge$ affiliation(≤ 1)] $\rightarrow$ [subject(≤ 3)]
DD_6	[title(≤ 2) $\wedge$ address(≤ 2) $\wedge$ affiliation(≤ 2)] $\rightarrow$ [subject(≤ 3)]
DD_7	[title(≤ 3) $\wedge$ author(≤ 3) $\wedge$ address(≤ 3) $\wedge$ affiliation(≤ 2)] $\rightarrow$ [subject(≤ 3)]
DD_8	[title(≤ 1) $\wedge$ affiliation(≤ 2)] $\rightarrow$ [subject(≤ 3)]
DD_9	[title(≤ 2) $\wedge$ address(≤ 1)] $\rightarrow$ [subject(≤ 3)]
DD_{10}	[title(≤ 3) $\wedge$ author($= 0$) $\wedge$ address(≤ 3)] $\rightarrow$ [subject(≤ 3)]
DD_{11}	[title(≤ 2) $\wedge$ address(≤ 1) $\wedge$ affiliation(≤ 1)] $\rightarrow$ [subject(≤ 2)]
DD_{12}	[title(≤ 1) $\wedge$ affiliation(≤ 1)] $\rightarrow$ [subject(≤ 2)]
DD_{13}	[title(≤ 1) $\wedge$ address(≤ 3) $\wedge$ affiliation(≤ 3)] $\rightarrow$ [subject(≤ 2)]
DD_{14}	[title(≤ 2) $\wedge$ author(≤ 3) $\wedge$ affiliation(≤ 3)] $\rightarrow$ [subject(≤ 2)]
DD_{15}	[title(≤ 1) $\wedge$ address(≤ 1)] $\rightarrow$ [subject(≤ 2)]
DD_{16}	[title($= 0$) $\wedge$ address(≤ 2)] $\rightarrow$ [subject(≤ 2)]
DD_{17}	[title(≤ 2) $\wedge$ author(≤ 3) $\wedge$ address(≤ 3)] $\rightarrow$ [subject(≤ 2)]

Table V. Cross Validation

Data size n	Result size	Exact to hold Δ_E	Max change to hold Δ_M	Total change to hold Δ_T
2000 (10%)	18	1 (0.055%)	6	51
6000 (30%)	20	1 (0.050%)	7	37
10000 (50%)	18	3 (0.166%)	2	18
14000 (70%)	16	6 (0.375%)	1	10
18000 (90%)	17	15 (0.882%)	1	2
20000 (100%)	17	–	–	–

sizes. Therefore, the constraints on author appear less in DDs in Table IV. Intuitively, we would like to evaluate how different the results are compared with those of 10000 tuples in Table III. Unfortunately, few works have been proposed to evaluate the quality of minimal cover of dependencies. Since the discovery result is a minimal cover, it is unlikely to enumerate all the DDs that can be implied by the minimal cover.

Nevertheless, we conduct a cross validation among the results of different data sizes, which roughly reflects the quality of minimal cover. Specifically, we consider 20000 tuples in total. For each result of n tuples, $\Sigma_n, n < 20000$, we study whether or how it holds in the remaining $20000 - n$ tuples. Since the discovered Σ_n is already known to hold in the first n tuples, it is equivalent to evaluate whether Σ_n holds in all 20000 tuples, that is, the difference between Σ_n and Σ_{20000}. Three criteria are employed to evaluate the difference between the results of n tuples and 20000 tuples.

—Δ_E, *Exact to hold.* Some DDs in Σ_n, for example, DD_2 in Table III, can be directly implied by the results of Σ_{20000}. We denote $\Delta_E(\Sigma_n, \Sigma_{20000})$ the number of DDs in the minimal cover Σ_n that exactly hold in Σ_{20000} without any change. For example, as illustrated in Table V, there are 3 DDs in Σ_{10000} that exactly hold in Σ_{20000}.

—Δ_M, *Max change to hold.* Although a DD in the minimal cover Σ_n might not exactly hold in Σ_{20000}, it does not indicate that other DDs implied by using this DD will not hold in Σ_{20000} either. For example, DD_{12} in Σ_{10000}, that is, [title(≤ 1) $\wedge$ affiliation(≤ 3)] $\rightarrow$ [subject(≤ 2)] in Table III, does not hold in Σ_{20000}. If we change the distance constraint affiliation(≤ 3) in DD_{12} to affiliation(≤ 1) as a new DD'_{12}, which can be implied by DD_{12}, then we can find that this DD'_{12} does hold in 20000 tuples, that is, can be implied by Σ_{20000}. Therefore, the quality of DDs appearing in the minimal cover cannot denote the quality of all the DDs that can be implied by the minimal cover. The implied DDs with a small change on distance constraint could hold in the 20000 tuples. We say the change of distance is 2, from affiliation(≤ 3) to affiliation(≤ 1). Consequently, for each DD_i in Σ_n, we can find a minimum change of distance $\Delta_C(\text{DD}_i, \Sigma_{20000})$ such that the changed DD$_i$ can be implied (hold) with respect to Σ_{20000}. If a DD$_i$ exactly holds without any change, then its $\Delta_C(\text{DD}_i, \Sigma_{20000}) = 0$. We may concern the maximum of change for a DD$_i$ in Σ_n, that is,

$$\Delta_M(\Sigma_n, \Sigma_{20000}) = \max_{\text{DD}_i \in \Sigma_n} \Delta_C(\text{DD}_i, \Sigma_{20000}).$$

For example, $\Delta_M = 2$ for 10000 tuples in Table V indicates that, for any DD$_i$ in Σ_{10000}, we only need a change of distance at most 2 in order to make it hold in 20000 tuples.

—Δ_T, *Total change to hold.* Finally, we report the total changes of distance for all DDs in Σ_n in order to make them hold in 20000 tuples.

$$\Delta_T(\Sigma_n, \Sigma_{20000}) = \sum_{\text{DD}_i \in \Sigma_n} \Delta_C(\text{DD}_i, \Sigma_{20000})$$

The total change of distance Δ_T roughly denotes the difference between two minimal covers Σ_n and Σ_{20000}. For example, $\Delta_T = 18$ for 10000 tuples in Table V denotes that with a total change of distance 18 on DDs in Σ_{10000}, we can ensure that the changed Σ_{10000} hold in the remaining (20000–10000) tuples.

Obviously, the lager Δ_E is preferred. However, the number Δ_E of DDs in the minimal cover does not denote the number of all the DDs that can be implied by the minimal cover and also hold in 20000 tuples. Intuitively, given more tuples in the instance, the discovered results are refined, for example, from affiliation($\leq$ 3) of DD$_{12}$ in Σ_{10000} to affiliation($\leq$ 1) of DD$_{12}$ in Σ_{20000}. Given affiliation($\leq$ 3) it is sufficient to imply DD$_{12}$ in 10000 tuples, while it requires restricting affiliation($\leq$ 1) for 20000 tuples. Indeed, smaller Δ_M and Δ_T of changes on distances roughly denote higher quality of Σ_n with respect to Σ_{20000}.

As the cross validation illustrated in Table V, the result sizes of discovered minimal covers are similar and not large under various data sizes. When the given instance is small, for example, 2000 tuples (10%) of 20000 tuples, only one DD in the minimal cover exactly holds without any change in the remaining 18000 tuples (90%). Again, we want to emphasize that the percentage of DDs in the minimal cover does not denote the proportion of all DDs implied by the minimal cover. Nevertheless, the changes of distances Δ_M and Δ_T are large between Σ_{2000} and Σ_{20000}, which roughly denotes that a large number of DDs holding in 2000 tuples might not be valid in 20000 tuples. With the increase of data sizes, more DDs in the discovered minimal cover (lager Δ_E) can exactly hold in 20000 tuples. Meanwhile, the changes of distances Δ_M and Δ_T become smaller. For example, when given 18000 tuples, a small change of distance 1 on 2 DDs in the discovered minimal cover will make all the implied DDs exactly hold in 20000 tuples.

6.4.2. Discovery Performance. We evaluate the cost of discovering CDKs and the minimal cover of DDs when given various relation instances (having 2000–20000 tuples) with different relation schemas R_1–R_6 (having 2–7 attributes). The relation instances select the first n ($n = 2000, \ldots, 20000$) tuples in the original order from the dataset. Different REDUCE algorithms are adopted, including the Brute-Force (BF) approach, the Bottom-Up search with Negative pruning (BU-Ne), Top-Down with Positive pruning (TD-Po), hybrid pruning (Hybrid), the Instance-Exclusion approach with positive pruning (IE-TD-Po), and hybrid pruning (IE-Hybrid).

Given various instance sizes $|I|$, Figures 5 and 6 report the time costs of discovering all CDKs relative to a given $\phi_1[R]$ and the minimal cover of all DDs that hold in the instance, respectively. First, all the approaches scale well with the number of tuples. Our techniques such as pruning can improve the performance and show slow time cost increase. It is not surprising according to our complexity analysis for REDUCE algorithms

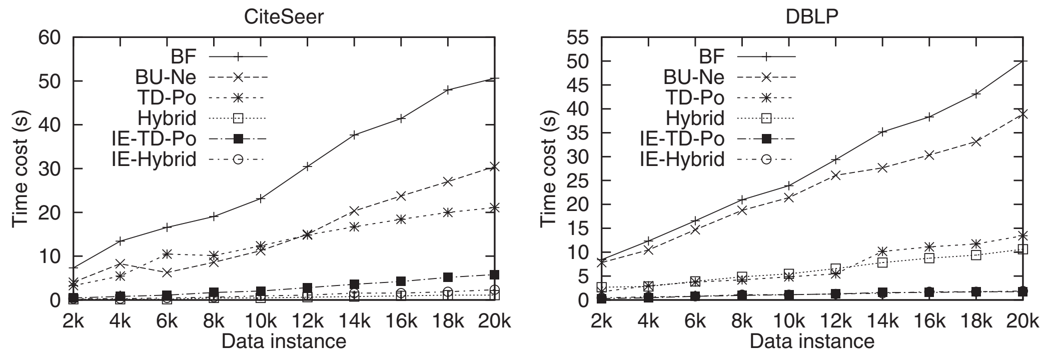

Fig. 5. CDKs discovery performance on various instance I.

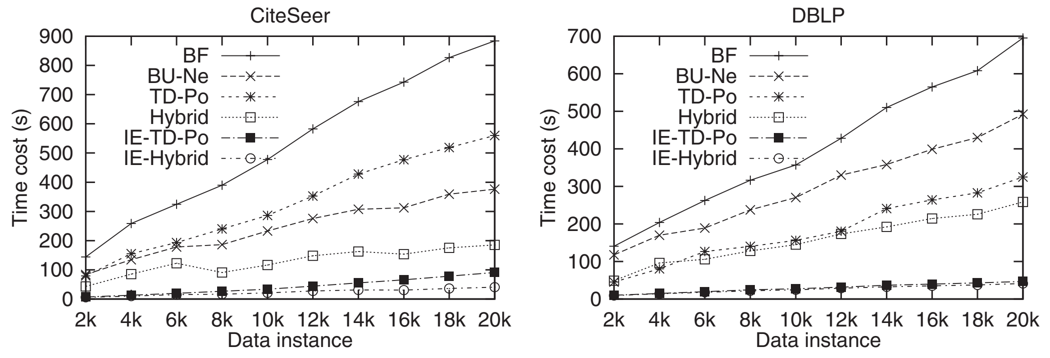

Fig. 6. DDs discovery performance on various instance I.

in Section 6.2, that is, $O(n^2)$ in terms of tuples $n = |I|$. It is worth noting that the pruning power may be affected by the features of data instances. For example, in the 4k of CiteSeer in Figure 5, BU-NE shows better performance than TD-PO, while TD-PO outperforms BU-NE in the 6k case. Similar results are also observed between TD-PO and HYBRID in DBLP in Figure 5. Moreover, due to the distinct features among different instances of the same dataset, the pruning power of a same approach may vary in different data sizes, for example, in 6k and 8k of the HYBRID approach in CiteSeer in Figure 6.

In the CiteSeer dataset in Figure 6, the BU-NE approach performs well, since most of candidate DDS do not hold in the given data instances. In this case, the TD-PO approach may fail, since it has to evaluate most of the candidates, that is, worst case. On the other hand, for example in the DBLP dataset in Figure 6, most of the candidates hold in the given instance, which favors by the TD-PO reduce. Therefore, the performance of TD-PO is significantly better than that of BU-NE. Nevertheless, the HYBRID approach can achieve low cost in both cases. Moreover, when TD-PO is valid, our IE-TD-PO exclusion can further improve the performance. Finally, as shown in Figures 5 and 6 IE-HYBRID can achieve the best performance and show significantly lower time cost than BF.

Figures 7 and 8 present the performance of discovery under various scheme sizes $|R|$. Unfortunately, the time costs of all the approaches increase exponentially with the increase of attributes. The results verify our hardness analysis of DDS and CDKS discovery in Section 6.2. As presented in Figures 7 and 8, although our pruning techniques can significantly improve performance, we cannot avoid the high complexity of

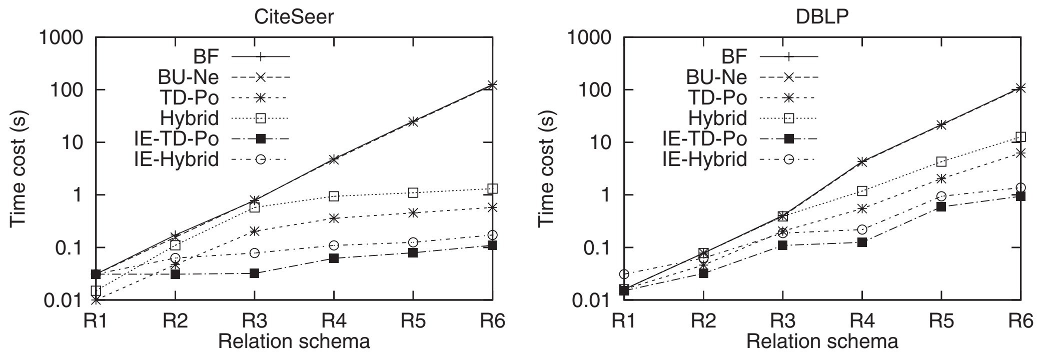

Fig. 7. CDKs discovery performance on various schema R.

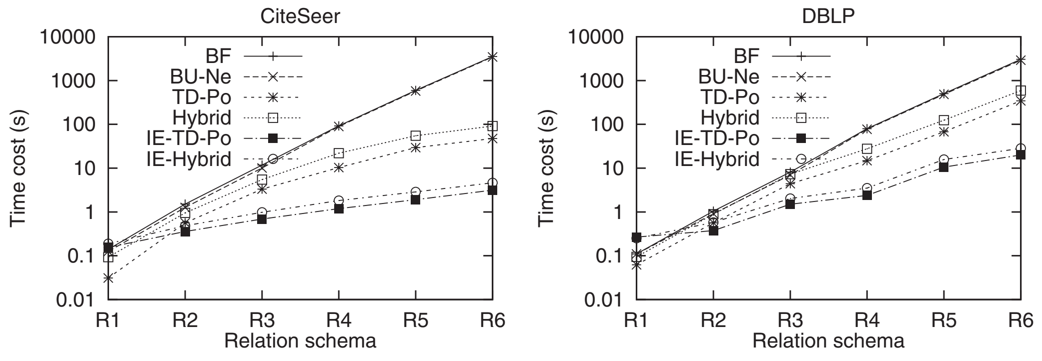

Fig. 8. DDs discovery performance on various schema R.

discovery. Nevertheless, the pruning techniques can achieve several orders of magnitude improvement compared with the BF approach. The instance-exclusion approaches, IE-TD-PO and IE-HYbrid, scale well especially on larger sizes of schema R.

To sum up, due to the inherent hardness, the discovery costs of DDS and CDKS are exponential in the number of attributes in relation $|R|$. Our advanced approaches such as IE-HYbrid can achieve 2–3 orders of magnitude improvement in discovering DDS and CDKS.

7. APPLICATIONS

In the following, we demonstrate several applications of DDS with real data evaluation. Dependencies used in the evaluation are also discovered by the discovery approaches in Section 6. Table VI lists the DDS in the Cora and Restaurant datasets that are discovered and used in the following application evaluation. For example, the DD$_4$ from Restaurant states that for any two tuples in an instance I of Restaurant, if their name and address values are similar, that is, with distances ≤ 4 and ≤ 5 respectively, then their id should be identified (denote the same entity in the real world). Moreover, another DD$_3$ indicates that if the address values of two tuples are highly similar with distance ≤ 1, even their name values are not so similar with distance ≤ 5, we can still imply the identification on id. In order to demonstrate the utility of differential dependencies, we illustrate several evaluations of applying these DDS in real applications.

Table VI. Examples of DDs in Cora and Restaurant

Cora	DD_1	$[\text{name}(\le 2) \wedge \text{title}(\le 3)] \rightarrow [\text{id}(=0)]$
	DD_2	$[\text{institution}(\le 2)] \rightarrow [\text{title}(=0) \wedge \text{venue}(=0)]$
Rest	DD_3	$[\text{name}(\le 5) \wedge \text{address}(\le 1)] \rightarrow [\text{id}(=0)]$
	DD_4	$[\text{name}(\le 4) \wedge \text{address}(\le 5)] \rightarrow [\text{id}(=0)]$
	DD_5	$[\text{name}(\le 2) \wedge \text{type}(\le 4)] \rightarrow [\text{address}(\le 6) \wedge \text{city}(\le 5)]$

Table VII. Example Instance of Empolyee

	name	institute	title	salary	ssn
t_1	John Depp	Tech. Univ.	Professor	60	111
t_2	J. Depp	Technical Univ.	Professor	60.2	111
t_3	J.C. Depp	Tech. University	Prof.	30	111
t_4	R. Depp	Western Univ.	Lecturer	30	222

7.1. Violation Detection

One of the most important applications of traditional FDs is to specify constraints of data. Those tuples that do not satisfy the given FDs are detected as violations to the constraints. Similarly, as a novel class of integrity constraints, DDs are naturally applicable for detecting violations.

7.1.1. Example. Consider the following DD in an Employee relation.

$$[\text{title}(\le 6)] \rightarrow [\text{salary}(\le 20)]$$

It states that the salary difference should not be large (i.e., in a same level) for two employees with similar title. Suppose that we have an instance of Employee relation, as presented in Table VII. Those tuples that do not follow the preceding DD constraint are detected as violations. Compared with traditional FDs based on equality function, DDs with differential function are more expressive with respect to integrity constraints. For example, t_1 and t_2 in Table VII are detected incorrectly as violations to a FD [title] $\rightarrow$ [salary], since they are identical on title but have different salary values. However, 60 and 60.2 denote a same salary level with different granularity and can be accepted by our DDs. On the other hand, t_2 and t_3 whose title edit distance is less than 6, have salary difference larger than 20, that is, a violation to the given DD. However, such violations are ignored by FDs which require equality on the left-hand side title. In fact, such violations cannot be detected by the extensions of FDs such as MFDs either.

Formally, given a set Σ of DDs, the violation detection is to find a minimum set of tuples that need to be removed from the relation instance I for Σ to hold in I. As presented in the following real data evaluation, DDs have better detection effectiveness than FDs.

Note that the FDs violation detection can be done efficiently in polynomial time, by grouping the tuples according to identical values [Arenas et al. 1999]. Unfortunately, the problem of detecting violations to DDs is hard. A reduction from the *vertex cover* problem can be developed. One can find a factor-2 approximation [Papadimitriou and Steiglitz 1982] by greedily removing both tuples of an instance of DDs violation. This approach ensures that all the violations can be detected, but some tuples may also be reported by approximation. The number of tuples reported which are not true violations is bounded by factor-2.

7.1.2. Evaluation. In the evaluation of this application, we randomly insert violations into the datasets. Both the values in the left-hand side and right-hand side attributes of data dependencies could be randomly replaced as violations. Let *truth* denote the set of tuples with inserted violations. Traditional FDs with equality function and our DDs with differential function are then applied to detect these violations. Let *found* be the set of detected violation tuples by FDs or DDs. We mainly study the detection accuracy,

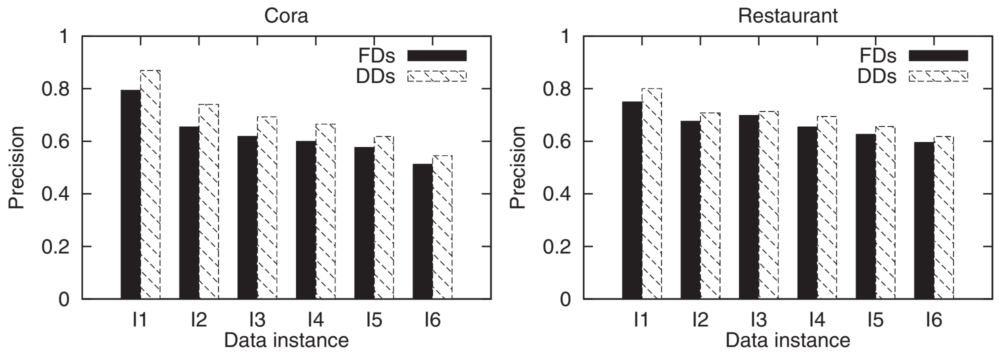

Fig. 9. Violation detection precision.

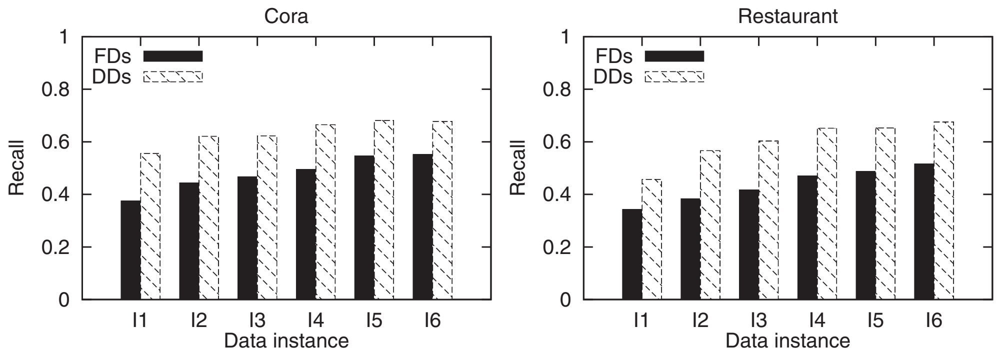

Fig. 10. Violation detection recall.

that is, detection $precision = \frac{truth \cap found}{found}$ and $recall = \frac{truth \cap found}{truth}$. The detection precision evaluates the portion of detected results that are the real violations we added, while detection recall reports the portion of all added violations which are detected. It is natural that higher detection precision and recall are preferred.

Figure 9 illustrates the detection precision by using dependencies with and without differential function. Since our DDs introduce differential functions in the right-hand side of dependencies in order to be tolerant to various information formats, the detection precision is higher than FDs with identical functions on Y. Moreover, DDs also introduce differential functions in the left-hand side, that is, we can address more tuples with violations. Therefore, as presented in Figure 10, the detection recall by using DDs is higher than FDs with equality function.

7.2. Data Partition

A typical partitioning task is to divide tuples in groups such that tuples in each group are similar to each other, for example, to partition tuples in groups where edit distance on each attribute is less than 10. We are not proposing a new partitioning method to improve the effectiveness. Instead, given a partitioning scheme on all the attributes of R, say $\phi_1[R]$, we perform the partitioning by using a Candidate Differential Key (CDK) $\phi_c[K]$ relative to $\phi_1[R]$ to improve the search efficiency.

7.2.1. Example. Consider a partitioning scheme over Employee.

$$\phi_1[R] = [\text{name}(\leq 3) \wedge \text{institute}(\leq 5) \wedge \text{title}(\leq 6) \wedge \text{salary}(\leq 40)]$$

That is, we have to divide the tuples into partitions, such that any two tuples in the same partition having name distance ≤ 3, institute distance ≤ 5, title distance ≤ 6, and salary distance ≤ 40. Suppose that we have a candidate differential key

$$\phi_c[K] = [\text{name}(\leq 3) \wedge \text{institute}(\leq 5)]$$

having $\phi_c[K] \rightarrow \phi_1[R]$ and $\phi_c[K] \succeq \phi_1[K]$. Instead of the long $\phi_1[R]$, we can equivalently use the short $\phi_c[K]$ as the partitioning scheme which has smaller cardinality with less computation cost.

As discussed, besides $\phi_1[R]$, a reduction can be found for an arbitrary differential function $\phi_R[Y]$, as a differential key. In order words, CDKs can also be applied in a group-by query.

In fact, semantic query optimization, which refers to the process of using integrity constraints (e.g., FDs) in order to optimize the evaluation of queries, has already been studied [Chakravarthy et al. 1990; Levy and Sagiv 1995]. As a novel type of integrity constraints, our DDs can naturally be applied in semantic query optimization as well.

For example, consider a group-by query on distance conditions.

SELECT * FROM Employee GROUP BY institute(≤ 5), title(≤ 6)

According to [institute(≤ 5)] $\rightarrow$ [institute(≤ 5) $\wedge$ title(≤ 6)], we can rewrite the query by using institute(≤ 5) only.

SELECT * FROM Employee GROUP BY institute(≤ 5)

7.2.2. Evaluation. In data partition evaluation, we mainly study the time performance of partitioning with and without CDKs. Specifically, let $\phi_1[R]$ be a partitioning scheme. For example, a partitioning scheme in Restaurant is [name(≤ 2) $\wedge$ address(≤ 6) $\wedge$ city(≤ 5) $\wedge$ type(≤ 4)]. By using DDs, we can find a left-reduced DDs which determines this partition scheme, that is, a CDK $\phi_c[K]$ relative to $\phi_1[R]$. For example, a reduced $\phi_c[K]$ = [name(≤ 2) $\wedge$ type(≤ 4)] can be a CDK in Restaurant such that $\phi_c[K] \rightarrow \phi_1[R]$ and $\phi_c[K] \succeq \phi_1[K]$. We evaluate the partitioning by using the original $\phi_1[R]$ and the equivalent reduced $\phi_c[K]$.

Figure 11 illustrates the time cost of reduced CDKs compared with original partitioning schemes. In the x-axis, each element a/b corresponds to a pair of original differential function for partitioning queries and its reduced CDK, where a denotes the cardinality of CDK and b denotes the cardinality of the original partition scheme, respectively. It is natural that the smaller the cardinality of a differential function, the lower the partition cost will be. Thereby, as presented in figures, the smaller the rate a/b, the more the performance is improved by using CDKs.

7.3. Record Linkage

Fan et al. [2009b] have proved the effectiveness in detecting duplicates, by utilizing Matching Dependencies (MDs) as matching rules. However, the original MDs associate exactly one similarity/distance constraint (a differential function) to each attribute. We argue that our DDs can address more matching rules by introducing different differential functions on one attribute.

7.3.1. Example. Consider a matching rule in Employee.

$$\text{DD}_1 \quad [\text{name}(\leq 5) \wedge \text{institute}(\leq 7)] \rightarrow [\text{ssn}(\rightleftharpoons)]$$

It states that for two records, whose name distance is less than 5, and institute distance is less than 7, they probably denote the same employee in the real world with identical[3] ssn. For example, according to the preceding DD$_1$, t_1 and t_2 in Table VII can be detected

[3]It could also be identified via update with dynamic semantics [Fan 2008; Fan et al. 2009b]. Without loss of generality, "identical" in the following also interpret the dynamic semantics via update.

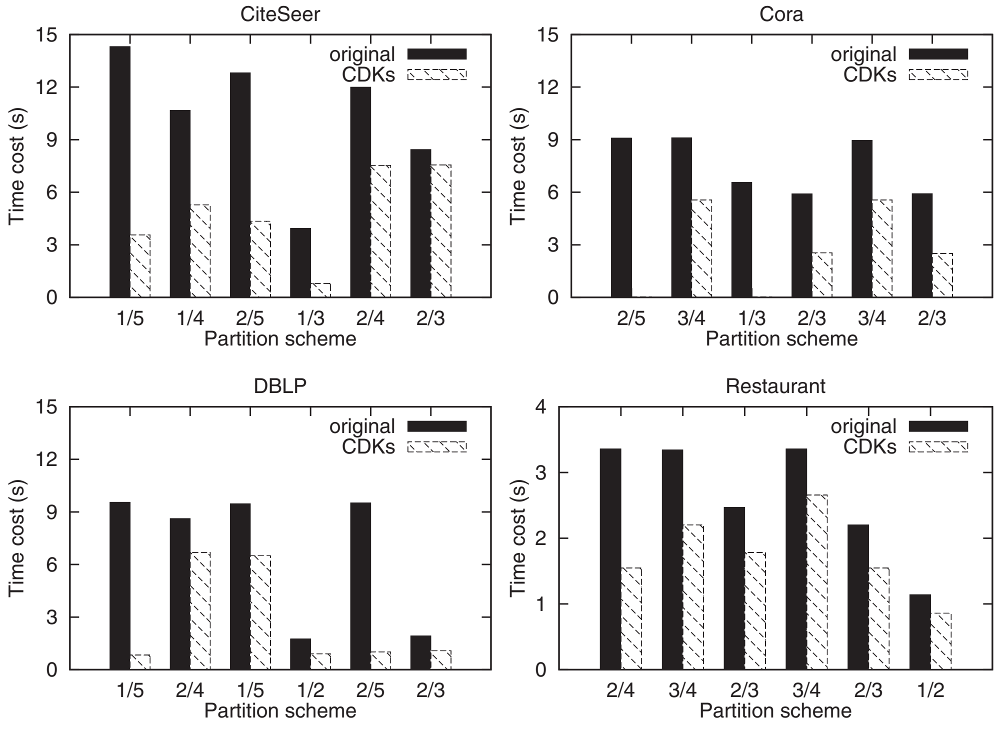

Fig. 11. Data partition performance.

as duplicates based on their distances on name(≤ 5) and institute(≤ 7), which is also the case in the real world, that is, having the identical ssn. It is notable that there is only one MD that can be declared on attributes name, institute, ssn, since each attribute has only one constraint.

By considering various differential functions, we may have another reasonable DD on the same attributes,

$$\text{DD}_2 \quad [\text{name}(\leq 3) \wedge \text{institute}(\leq 15)] \rightarrow [\text{ssn}(\rightleftharpoons)],$$

which is also a valid matching rule. According to DD_2, t_2 and t_3 can be detected as duplicates which are not detected by using DD_1.

7.3.2. Evaluation. Restaurant and Cora are two datasets collected for record linkage purposes, where two tuples with identical id denotes duplicates in the real world. A typical matching rule, for example [name(≤ 5) $\wedge$ address(≤ 1)] $\rightarrow$ [id($= 0$)] in Restaurant, is used to identify those duplicate tuples according to the left-hand side attributes. We mainly observe the accuracy of returned results, that is, precision and recall in Figure 12 and 13 respectively, where *truth* is the set of tuple pairs in a given instance that are true duplicates and *found* is the set of tuple pairs in the returned results.

Our DDs are compared with the previous MDs as matching rules. As illustrated in Figure 12, DDs have comparable precision to MDs, since both MDs and DDs are valid matching rules. On the other hand, instead of associating only one differential function on each attribute in MDs, our DDs can specify various differential functions on the same attribute for different matching rules. By addressing more matching rules, our DDs can return more answers than MDs. Consequently, as illustrated in Figure 13, the recall of

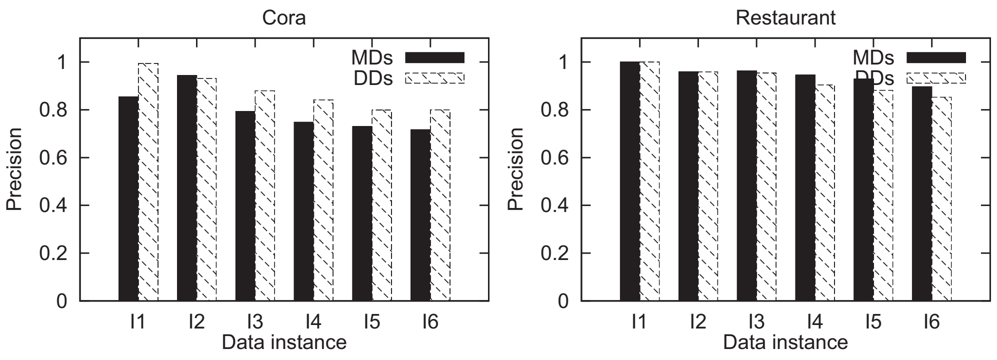

Fig. 12. Record linkage precision.

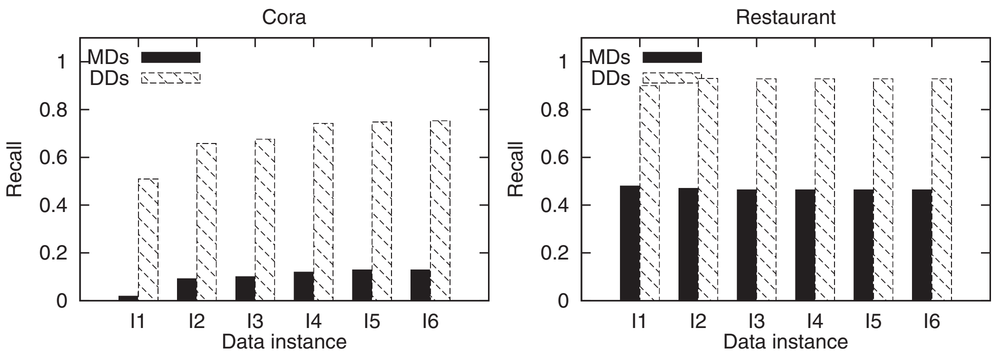

Fig. 13. Record linkage recall.

DDS is higher than that of MDS. Moreover, since the Restaurant dataset is cleaner, the precision and recall of both DDS and MDS are higher than those in the Cora dataset.

8. CONCLUSIONS

In this article, we propose a novel form of Differential Dependencies (DDS), which specify constraints on difference semantics. To our best knowledge, this is the first work on addressing both the theoretical and practical issues of differential dependencies between attributes. We give the definitions and properties of DDS and differential keys, and investigate reasoning about DDS. A sound and complete inference system is also presented for DDS, which raises the problem of finding the minimal cover for a DDS set. Algorithms for discovering DDS and differential keys are then developed and evaluated in real data experiments. Finally, we also illustrate the details and read data evaluation of several applications of DDS.

Future Work. In real practice, it is often the case that data dependencies may "almost" hold in a data instance, which are known as approximate dependencies, such as as approximate functional dependencies [Huhtala et al. 1998, 1999]. As a natural extension in the future work, we can study *approximate differential dependencies* as well. The error measure [Kivinen and Mannila 1995] can also be adopted to evaluate how a DD "almost" holds in a data instance, that is, the minimum number of tuples that have to be removed from the data instance for the given dependency to hold. Unfortunately, the computation of error measure (as well as the validation problem of approximate DDS) seems to be hard. The transitivity cannot be assumed, that is, from $(t_1, t_2) \asymp \phi[X]$ and

$(t_2, t_3) \asymp \phi[X]$ it does not necessarily follow that $(t_1, t_3) \asymp \phi[X]$. The efficient validation computation based on disjoint grouping [Huhtala et al. 1999] cannot be applied in this case of differential functions on various attributes. Therefore, in the future work, efficient (approximation) algorithms are planned to be developed for computing the error measures of DDs. Besides the error measure, we may also define other measures of dependencies, for example, on how many tuple pairs supporting a dependency. Instead of finding a minimal cover of all the dependencies holding in a data instance, we may return only a subset of dependencies with the highest (top-k) measures, that is, the ranking problem of dependencies. In the current work, we mainly focus on the minimal cover of all dependencies holding in a data instance, which could be considered as a set of candidates for possible ranking.

We believe that various interesting applications may be developed upon the proposed notation of differential dependencies. For example, successful conditioning of dependencies [Bohannon et al. 2007; Fan et al. 2008a] may also be highlighted for DDs, that is, to make it hold in a subset of tuples instead of the entire table. It is worth noting that Conditional Functional Dependencies (CFDs) and Differential Dependencies (DDs) are two different directions of extensions on FDs. In particular, CFDs employ conditions in the dependency, while the equality function is still used. On the other hand, instead of equality function, our DDs introduce the differential function in the dependency. Consequently, as a future work, these two exertions may also cooperate together, that is, extending DDs with conditions. Moreover, in the aspect of application, data repairing with DDs is promising yet challenging, since it is already hard with dependencies on equality [Chomicki and Marcinkowski 2005]. Furthermore, DDs can also be utilized as integrity rules in *dataspaces* [Franklin et al. 2005], where data dependencies should be tolerant to various information formats of heterogeneous data.

REFERENCES

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

Arenas, M., Bertossi, L. E., and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99)*. 68–79.

Armstrong, W. W. 1974. Dependency structures of data base relationships. In *Proceedings of the IFIP Congress*. 580–583.

Aspvall, B., Plass, M. F., and Tarjan, R. E. 1979. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett. 8*, 3, 121–123.

Bitton, D., Millman, J., and Torgersen, S. 1989. A feasibility and performance study of dependency inference. In *Proceedings of the International Conference on Data Engineering (ICDE'89)*. 635–641.

Bohannon, P., Fan, W., Geert S. F., Jia, X., and Kementsietsidis, A. 2007. Conditional functional dependencies for data cleaning. In *Proceedings of the International Conference on Data Engineering (ICDE'07)*. 746–755.

Bravo, L., Fan, W., Geerts, F., and Ma, S. 2008. Increasing the expressivity of conditional functional dependencies without extra complexity. In *Proceedings of the International Conference on Data Engineering (ICDE'08)*. 516–525.

Bravo, L., Fan, W., and Ma, S. 2007. Extending dependencies with conditions. In *Proceedings of the International Conference on Very Large Databases (VLDB'07)*. 243–254.

Chakravarthy, U. S., Grant, J., and Minker, J. 1990. Logic-Based approach to semantic query optimization. *ACM Trans. Datab. Syst. 15*, 2, 162–207.

Chaudhuri , S., Sarma, A. D., Ganti, V., and Kaushik, R. 2007. Leveraging aggregate constraints for deduplication. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 437–448.

Chazelle, B. and Guibas, L. J. 1986a. Fractional cascading: I. A data structuring technique. *Algorithmica 1*, 2, 133–162.

Chazelle, B. and Guibas, L. J. 1986b. Fractional cascading: Ii. Applications. *Algorithmica 1*, 2, 163–191.

Chiang, F. and Miller, R. J. 2008. Discovering data quality rules. *Proc. VLDB 1*, 1, 1166–1177.

CHOMICKI, J. AND MARCINKOWSKI, J. 2005. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*. 119–150.

CONG, G., FAN, W., GEERTS, F., JIA, X., AND MA, S. 2007. Improving data quality: Consistency and accuracy. In *Proceedings of the International Conference on Very Large Databases (VLDB'07)*. 315–326.

DE BERG, M., CHEONG, O., VAN KREVELD, M., AND OVERMARS, M. 2008. *Computational Geometry: Algorithms and Applications* 3rd Ed. Springer.

DONG, J. AND HULL, R. 1982. Applying approximate order dependency to reduce indexing space. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 119–127.

EL MAGARMID, A. K., IPEIROTIS, P. G., AND VERYKIOS, V. S. 2007. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Engin. 19*, 1, 1–16.

FAN, W. 2008. Dependencies revisited for improving data quality. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'08)*. 159–170.

FAN, W., GEERTS, F., JIA, X., AND KEMENTSIETSIDIS, A. 2008a. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Datab. Syst. 33*, 2.

FAN, W., GEERTS, F., LAKSHMANAN, L. V. S., AND XIONG, M. 2009a. Discovering conditional functional dependencies. In *Proceedings of the International Conference on Data Engineering (ICDE'09)*. 1231–1234.

FAN, W., LI, J., JIA, X., AND MA, S. 2009b. Reasoning about record matching rules. In *Proceedings of the International Conference on Very Large Databases (VLDB'09)*.

FAN, W., MA, S., HU, Y., LIU, J., AND WU, Y. 2008b. Propagating functional dependencies with conditions. *Proc. VLDB 1*, 1, 391–407.

FLACH, P. A. AND SAVNIK, I. 1999. Database dependency discovery: A machine learning approach. *Artif. Intell. Comm. 12*, 3, 139–160.

FRANKLIN, M. J., HALEVY, A. Y., AND MAIER, D. 2005. From databases to dataspaces: A new abstraction for information management. *SIGMOD Rec. 34*, 4, 27–33.

GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

GINSBURG, S. AND HULL, R. 1983a. Order dependency in the relational model. *Theor. Comput. Sci. 26,* 149–195.

GINSBURG, S. AND HULL, R. 1983b. Sort sets in the relational model. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'83)*. 332–339.

GINSBURG, S. AND HULL, R. 1986. Sort sets in the relational model. *J. ACM 33,* 3, 465–488.

GOLAB, L., KARLOFF, H. J., KORN, F., SAHA, A., AND SRIVASTAVA, D. 2009. Sequential dependencies. *Proc. VLDB 2,* 1, 574–585.

GOLAB, L., KARLOFF, H. J., KORN, F., SRIVASTAVA, D., AND YU, B. 2008. On generating near-optimal tableaux for conditional functional dependencies. *Proc. VLDB 1*, 1, 376–390.

GRAVANO, L., IPEIROTIS, P. G., JAGADISH, H. V., KOUDAS, N., MUTHUKRISHNAN, S., PIETARINEN, L., AND SRIVASTAVA, D. 2001. Using q-grams in a dbms for approximate string processing. *IEEE Data Engin. Bull. 24*, 4, 28–34.

HERNANDEZ, M. A. AND STOLFO, S. J. 1995. The merge/purge problem for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 127–138.

HUHTALA, Y., KARKKAINEN, J., PORKKA, P., AND TOIVONEN, H. 1998. Efficient discovery of functional and approximate dependencies using partitions. In *Proceedings of the International Conference on Data Engineering (ICDE'98)*. 392–401.

HUHTALA, Y., KARKKAINEN, J., PORKKA, P., AND TOIVONEN, H. 1999. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J. 42*, 2, 100–111.

ILYAS, I . F., MARKL, V., HAAS, P. J., BROWN, P., AND ABOULNAGA, A. 2004. Cords: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 647–658.

KARP, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations,* R. E. Miller and J. W. Thatcher Eds., Plenum Press, 85–103.

KIMURA, H., HUO, G., RASIN, A., MADDEN, S., AND ZDONIK, S. B. 2009. Correlation maps: A compressed access method for exploiting soft functional dependencies. *Proc. VLDB 2,* 1, 1222–1233.

KIVINEN, J. AND MANNILA, H. 1995. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci. 149*, 1, 129–149.

KORN, F., MUTHUKRISHNAN, S., AND ZHU, Y. 2003. Checks and balances: Monitoring data quality problems in network traffic databases. In *Proceedings of the International Conference on Very Large Databases (VLDB'03)*. 536–547.

KOUDAS, N., SAHA, A., SRIVASTAVA, D., AND VENKATASUBRAMANIAN, S. 2009. Metric functional dependencies. In *Proceedings of the International Conference on Data Engineering (ICDE'09)*. 1275–1278.

KRAMER, S. AND PFAHRINGER, B. 1996. Efficient search for strong partial determinations. In *Proceedings of the International SIGKDD Conference on Knowledge Discovery and Data Mining.* 371–374.

LEVENE, M. AND LOIZOU, G. 1999. *A Guided Tour of Relational Databases and Beyond*. Springer.

LEVY, A. Y. AND SAGIV, Y. 1995. Semantic query optimization in datalog programs. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'95).* 163–173.

MANNILA, H. AND RAI HA, K.-J. 1987. Dependency inference. In *Proceedings of the International Conference on Very Large Databases (VLDB'87).* 155–158.

MANNILA, H. AND RAI HA, K.-J. 1989. Practical algorithms for finding prime attributes and testing normal forms. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'89).* 128–133.

MANNILA, H. AND RAI HA, K.-J. 1992. *Design of Relational Databases*. Addison-Wesley.

MANNILA, H. AND RAI HA, K.-J. 1994. Algorithms for inferring functional dependencies from relations. *Data Knowl. Engin. 12*, 1, 83–99.

NAVARRO, G. 2001. A guided tour to approximate string matching. *ACM Comput. Surv. 33*, 1, 31–88.

NG, W. 1999. Ordered functional dependencies in relational databases. *Inf. Syst. 24*, 7, 535–554.

NG, W. 2001. An extension of the relational data model to incorporate ordered domains. *ACM Trans. Datab. Syst. 26*, 3, 344–383.

PAPADIMITRIOU, C. H. AND STEIGLITZ, K. 1982. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall.

SCHLIMMER, J. C. 1993. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *Proceedings of the International Conference on Machine Learning (ICML'93).* 284–290.

WIJSEN, J. 1998. Reasoning about qualitative trends in databases. *Inf. Syst. 23*, 7, 463–487.

WIJSEN, J. 2001. Trends in databases: Reasoning and mining. *IEEE Trans. Knowl. Data Engin. 13*, 3, 426–438.

WYSS, C. M., GIANNELLA, C., AND ROBERTSON, E. L. 2001. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances - extended abstract. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DaWaK'01).* 101–110.

Received August 2010; revised January 2011; accepted March 2011