Discovering Conditional Functional Dependencies

Wenfei Fan [1,2], Floris Geerts [1], Laks V.S. Lakshmanan [3], Ming Xiong [2]

[1]*University of Edinburgh, UK* [2]*Bell Laboratories, USA* [3]*University of British Columbia, Canada*
{wenfei,fgeerts}@inf.ed.ac.uk {wenfei,xiong}@research.bell-labs.com laks@cs.ubc.ca

***Abstract*— This paper investigates the discovery of conditional functional dependencies (CFDs). CFDs are a recent extension of functional dependencies (FDs) by supporting patterns of semantically related constants, and can be used as rules for cleaning relational data. However, finding CFDs is an expensive process that involves intensive manual effort. To effectively identify data cleaning rules, we develop techniques for discovering CFDs from sample relations. We provide three methods for CFD discovery. The first, referred to as CFDMiner, is based on techniques for mining closed itemsets, and is used to discover *constant* CFDs, namely, CFDs with constant patterns only. The other two algorithms are developed for discovering general CFDs. The first algorithm, referred to as CTANE, is a levelwise algorithm that extends TANE, a well-known algorithm for mining FDs. The other, referred to as FastCFD, is based on the depth-first approach used in FastFD, a method for discovering FDs. It leverages closed-itemset mining to reduce search space. Our experimental results demonstrate the following. (a) CFDMiner can be multiple orders of magnitude faster than CTANE and FastCFD for constant CFD discovery. (b) CTANE works well when a given sample relation is large, but it does not scale well with the arity of the relation. (c) FastCFD is far more efficient than CTANE when the arity of the relation is large.**

I. Introduction

Conditional functional dependencies (CFDs) [1] were recently introduced for data cleaning. They extend standard functional dependencies (FDs) by enforcing patterns of semantically related constants. CFDs have been proven more effective than FDs in detecting and repairing inconsistencies (dirtiness) of data [1], [2], and are expected to be adopted by data-cleaning tools that currently employ standard FDs (*e.g.,* [3], [4], [5]; see [6], [7] for a survey of such tools).

However, for CFD-based cleaning methods to be effective in practice, it is necessary to have techniques in place that can *automatically discover*, *profile*, or *learn* CFDs from sample data, to be used as data cleaning rules. As indicated in [8], profiling of data cleaning rules is critical to commercial data quality tools.

This practical concern highlights the need for studying the *discovery problem* for CFDs: given a sample instance r of a relation schema R, it is to find a *canonical cover* of all CFDs that hold on r, *i.e.,* a set of CFDs that is logically equivalent to the set of all CFDs that hold on r. To reduce redundancy, each CFD in the canonical cover should be *minimal*, *i.e., nontrivial* and *left-reduced* (see [9] for nontrivial and left-reduced FDs).

The discovery problem is, however, *highly nontrivial*. It is already hard for traditional FDs since, among other things, a canonical cover of FDs discovered from a relation r is inherently *exponential* in the *arity* of the schema of r, *i.e.,* the number of attributes in R. Since CFD discovery subsumes FD discovery, *the exponential complexity* carries over to CFD discovery. Moreover, CFD discovery requires mining of semantic patterns with constants, a challenge that was not encountered when discovering FDs, as illustrated by the example below.

Example 1: The following relation schema cust is taken from [1]. It specifies a customer in terms of the customer's phone (country code (CC), area code (AC), phone number (PN)), name (NM), and address (street (STR), city (CT), zip code (ZIP)). An instance r_0 of cust is shown in Fig. 1.

Traditional FDs that hold on r_0 include the following:

f_1: $[\mathsf{CC}, \mathsf{AC}] \rightarrow \mathsf{CT}$
f_2: $[\mathsf{CC}, \mathsf{AC}, \mathsf{PN}] \rightarrow \mathsf{STR}$

Here f_1 requires that two customers with the same country- and area-codes also have the same city; similarly for f_2.

In contrast, the CFDs that hold on r_0 include not only the FDs f_1 and f_2, *but also* the following (and more):

ϕ_0: ($[\mathsf{CC}, \mathsf{ZIP}] \rightarrow \mathsf{STR}$, (44, _ || _))
ϕ_1: ($[\mathsf{CC}, \mathsf{AC}] \rightarrow \mathsf{CT}$, (01, 908 || MH))
ϕ_2: ($[\mathsf{CC}, \mathsf{AC}] \rightarrow \mathsf{CT}$, (44, 131 || EDI))
ϕ_3: ($[\mathsf{CC}, \mathsf{AC}] \rightarrow \mathsf{CT}$, (01, 212 || NYC))

In ϕ_0, (44, _ || _) is the pattern tuple that enforces a binding of semantically related constants for attributes (CC, ZIP, STR) in a tuple. It states that for customers in the UK, ZIP uniquely determines STR. It is an FD that only holds on the subset of tuples with the pattern "CC = 44", rather than on the entire relation r_0. CFD ϕ_1 assures that for any customer in the US (country code 01) with area code 908, the city of the customer *must* be MH, as enforced by its pattern tuple (01, 908 || MH); similarly for ϕ_2 and ϕ_3. These cannot be expressed as FDs.

More specifically, a CFD is of the form $(X \rightarrow A, t_p)$, where $X \rightarrow A$ is an FD and t_p is a *pattern tuple* with attributes in X and A. The pattern tuple consists of constants and an unnamed variable '_' that matches an arbitrary value. To discover a CFD it is necessary to find not only the traditional FD $X \rightarrow A$ but also its pattern tuple t_p. With the same FD $X \rightarrow A$ there are possibly multiple CFDs defined with different pattern tuples, *e.g.,* ϕ_1–ϕ_3. Hence a canonical cover of CFDs that hold on r_0 is typically much larger than its FD counterpart. Indeed, as recently shown by [10], provided that a fixed FD $X \rightarrow A$ is already given, the problem for discovering sensible patterns associated with the FD alone is already NP-complete. □

Prior work. The discovery problem has been studied for FDs for two decades [11], [12], [13], [14], [15], [16], [17], [18], [19] for database design, data archiving, OLAP and data mining. It was first investigated in [13], which shows that the problem is inherently exponential in the arity $|R|$ of the schema R of sample data r. One of the best-known methods

1084-4627/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDE.2009.208

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202

Fig. 1. An instance r_0 of the cust relation.

for FD discovery is TANE [14], a *levelwise* algorithm that searches an attribute-set containment lattice and derives FDs with $\ell + 1$ attributes from sets of ℓ attributes, with pruning based on FDs generated in previous levels. TANE takes linear time in the size $|r|$ of *input* sample r, and works well when the arity $|R|$ is not very large. Another algorithm, referred to as FastFD [15], explores the connection between FD discovery and the problem of finding minimal covers of hypergraphs, and employs the depth-first strategy to search minimal covers. It takes (almost) linear-time in the size of the *output*, *i.e.,* in the size of the FD cover. It scales better than TANE when the arity is large, but it is more sensitive to the size $|r|$ (it is in $O(|r|^2 \log |r|)$ time, data complexity).

Recently two sets of algorithms have been developed for discovering CFDs [10], [20]. For a fixed traditional FD fd, [10] showed that it is NP-complete to find useful patterns that, together with fd, make high quality CFDs. They provide efficient heuristic algorithms for discovering patterns from samples *w.r.t.* a fixed FD. An algorithm for discovering CFDs, including both traditional FDs and their associated patterns, was presented in [20], which is an extension of TANE. The CFDs discovered by the TANE extensions may, however, contain redundant patterns, as elaborated in [21].

Constant CFD discovery is related to association rule mining (*e.g.,* [22]) and in particular, closed and free itemsets mining (*e.g.,* [23], [24]). There is an intimate connection between left-reduced constant CFDs and non-redundant association rules, which can be found from closed and free itemsets.

Contributions. The key contributions of our work include the following. **(1)** We propose a notion of minimal (frequent) CFDs based on both the minimality of attributes and the minimality of patterns. **(2)** Our first algorithm, CFDMiner, discovers constant CFDs. It explores the connection between minimal constant CFDs and closed and free itemsets. **(3)** Our second algorithm, CTANE, extends TANE to discover general CFDs based on an attribute-set/pattern tuple lattice. **(4)** Our third algorithm, FastCFD, discovers general CFDs. It hereby employs a depth-first search strategy instead of following the levelwise approach. A novel pruning technique is introduced by leveraging constant CFDs found by CFDMiner. **(5)** Our final contribution is an experimental study of the effectiveness and efficiency of our algorithms, based on real-life data and synthetic datasets.

Organization. Section II reviews CFDs, defines minimal and frequent CFDs, and states the discovery problem. Section III presents CFDMiner, CTANE, FastCFD, and a summary of our experimental results. Finally, Section IV concludes our work.

II. CFDs AND CFD DISCOVERY

In this section we first review the definition of CFDs [1]. We then formalize the notions of minimal CFDs and frequent CFDs. Finally, we state the discovery problem for CFDs.

A. Conditional Functional Dependencies

Consider a relation schema R defined over a fixed set of attributes, denoted by $\mathbf{attr}(R)$. For each attribute $A \in \mathbf{attr}(R)$, we use $\mathbf{dom}(A)$ to denote its domain.

CFDs. A *conditional functional dependency* (CFD) φ over R is a pair $(X \rightarrow A,\ t_p)$, where (1) X is a set of attributes in $\mathbf{attr}(R)$, and A is a single attribute in $\mathbf{attr}(R)$, (2) $X \rightarrow A$ is a standard FD, referred to as the FD *embedded in* φ; and (3) 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 $\mathbf{dom}(B)$, or an unnamed variable '_' that draws values from $\mathbf{dom}(B)$.

We denote X as LHS(φ) and A as RHS(φ). If A also occurs in X, we use A_L and A_R to indicate the occurrence of A in the LHS(φ) and RHS(φ), respectively. We separate the attributes in X and A in a pattern tuple with ' $\|$ '.

Standard FDs are a special case of CFDs. Indeed, an FD $X \rightarrow A$ can be expressed as a CFD $(X \rightarrow A,\ t_p)$, where $t_p[B] =$ _ for each B in $X \cup \{A\}$.

Semantics. To give the semantics of CFDs, we define an order $\leq$ on constants and the unnamed variable '_': $\eta_1 \leq \eta_2$ if either $\eta_1 = \eta_2$, or η_1 is a constant a and η_2 is '_'.

The order $\leq$ naturally extends to tuples, *e.g.,* (44, "EH4 1DT", "EDI") $\leq$ (44, _ , _) but (01, 07974, "Tree Ave.") $\not\leq$ (44, _ , _). We say that a tuple t_1 *matches* t_2 if $t_1 \leq t_2$. We write $t_1 \ll t_2$ if $t_1 \leq t_2$ but $t_2 \not\leq t_1$, *i.e.,* when t_2 is "more general" than t_1. For instance, (44, "EH4 1DT", "EDI") $\ll$ (44, _, _).

An instance r of R *satisfies* the CFD φ (or φ *holds* on r), denoted by $r \models \varphi$, iff for *each pair* of tuples t_1, t_2 in r, if $t_1[X] = t_2[X] \leq t_p[X]$ then $t_1[A] = t_2[A] \leq t_p[A]$.

Intuitively, φ is a constraint defined on the set $r_\varphi = \{t \mid t \in r, t[X] \leq t_p[X]\}$ such that for any $t_1, t_2 \in r_\varphi$, if $t_1[X] = t_2[X]$, then (a) $t_1[A] = t_2[A]$, and (b) $t_1[A] \leq t_p[A]$. Here (a) enforces the semantics of the embedded FD on the set r_φ, and (b) assures the binding between *constants* in $t_p[A]$ and *constants* in $t_1[A]$. That is, φ constrains *the subset* r_φ of r identified by $t_p[X]$, rather than the *entire* instance r.

Example 2: The instance r_0 of Fig. 1 satisfies CFDs f_1, f_2 and ϕ_0–ϕ_3 of Example 1. It does *not* satisfy the CFD ψ = ([CC, ZIP] $\rightarrow$ STR, (_ , _ $\|$ _)). Indeed, t_1 and t_4 *violate* ψ since t_1[CC, ZIP] = t_4[CC,ZIP] $\leq$ (_ , _), but t_1[STR] $\neq$ t_4[STR]. Nor does r satisfy ψ' = (AC $\rightarrow$ CT, (131 $\|$ EDI)) since t_8 violates ψ': t_8[AC] $\leq$ (131) but t_8[CT] $\not\leq$ (EDI). From this one can see that while *two* tuples are needed to violate an FD, CFDs can be violated by a *single* tuple. □

We say that an instance r of R *satisfies* a set Σ of CFDs over R, denoted by $r \models \Sigma$, if $r \models \varphi$ for *each* CFD $\varphi \in \Sigma$.

For two sets Σ and Σ' of CFDs defined over the same schema R, we say that Σ is *equivalent to* Σ', denoted by $\Sigma \equiv \Sigma'$, iff for any instance r of R, $r \models \Sigma$ iff $r \models \Sigma'$.

Classification of CFDs. A CFD $(X \rightarrow A,\ t_p)$ is called a *constant* CFD if its pattern tuple t_p consists of constants only,

i.e., $t_p[A]$ is a constant and for all $B \in X$, $t_p[B]$ is a constant. It is called a *variable* CFD if $t_p[A] = \_$, *i.e.,* the RHS of its pattern tuple is the unnamed variable '_'.

B. The Discovery Problem for CFDs

Below we first formalize the notions of minimal CFDs and frequent CFDs. We then state the discovery problem for CFDs.

Minimal CFDs. A CFD $\varphi = (X \to A,\ t_p)$ over R is said to be *trivial* if $A \in X$. If φ is trivial, then either it is satisfied by all instances of R (*e.g.,* when $t_p[A_L] = t_p[A_R]$), or it is satisfied by none of the instances in which there is a tuple t such that $t[X] \leq t_p[X]$ (*e.g.,* if $t_p[A_L]$ and $t_p[A_R]$ are distinct constants). In the sequel we consider nontrivial CFDs only.

A constant CFD $(X \to A,\ (t_p \parallel a))$ is said to be *left-reduced* on r if for any $Y \subsetneq X$, $r \not\models (Y \to A, (t_p[Y] \parallel a))$.

A variable CFD $(X \to A,\ (t_p \parallel \_))$ is *left-reduced* on r if (1) $r \not\models (Y \to A, (t_p[Y] \parallel \_))$ for any proper subset $Y \subsetneq X$, and (2) $r \not\models (X \to A, (t'_p[X] \parallel \_))$ for any t'_p with $t_p \ll t'_p$.

Intuitively, these assure the following: (1) none of its LHS attributes can be removed, *i.e.,* the minimality of attributes, and (2) none of the constants in its LHS pattern can be "upgraded" to '_', *i.e.,* the pattern t_p is "most general", or in other words, it assures the minimality of patterns.

A *minimal* CFD φ *on* r is a nontrivial, left-reduced CFD such that $r \models \varphi$. Intuitively, a minimal CFD is non-redundant.

Example 3: On the sample r_0 of Fig. 1, ϕ_2 of Example 1 is a minimal constant CFDs, and f_1, f_2 and ϕ_0 are minimal variable CFDs. However, ϕ_3 is not minimal: if we drop CC from LHS(ϕ_3), r_0 still satisfies (AC $\to$ CT, (212 $\parallel$ NYC)) since there is only one tuple (t_3) with AC = 212 in r_0. Similarly, ϕ_1 is not minimal since CC can be dropped. □

Frequent CFDs. The *support* of a CFD $\varphi = (X \to A,\ t_p)$ in r, denoted by $\sup(\varphi, r)$, is defined to be the set of tuples t in r such that $t[X] \leq t_p[X]$ and $t[A] \leq t_p[A]$, *i.e.,* tuples that match the pattern of φ. For a natural number $k \geq 1$, a CFD φ is said to be *k-frequent* in r if $\sup(\varphi, r) \geq k$. For instance, ϕ_1, ϕ_2 of Example 1 are 3-frequent and 2-frequent, respectively. Moreover, f_1, f_2 are 8-frequent.

Problem statement. A *canonical cover* of CFDs on r *w.r.t.* k is a set Σ of minimal, k-frequent CFDs in r, such that Σ is equivalent to the set of all k-frequent CFDs that hold on r. Our CFD *discovery problem* is to find a canonical cover of CFDs on r *w.r.t.* k.

III. Discovering CFDs: Algorithms and Results

In this section, we present high-level descriptions of our algorithms for CFD profiling and a summary of our experimental results. Readers are referred to [21] for more details.

A. CFDMiner: Discovering Constant CFDs

Given an instance r of R and a support threshold k, our algorithm for constant CFD profiling, i.e., CFDMiner, finds a canonical cover of k-frequent minimal constant CFDs of the form $(X \to A, (t_p \parallel a))$.

Free and closed itemsets. An itemset is a pair (X, t_p), where $X \subseteq \mathbf{attr}(R)$ and t_p is a constant pattern over X. The *support* of (X, t_p) in an instance r, denoted by $\mathsf{supp}(X, t_p, r)$, is defined as the set of tuples in r that match with t_p on the X-attributes. We say that (Y, s_p) is more general than (X, t_p), denoted by $(X, t_p) \preceq (Y, s_p)$, if $Y \subseteq X$ and $t_p[Y] = s_p$. Clearly, if $(X, t_p) \preceq (Y, s_p)$ then $\mathsf{supp}(X, t_p, r) \subseteq \mathsf{supp}(Y, s_p, r)$. For a natural number $k \geq 1$, we say that an itemset (X, t_p) is k-frequent if $|\mathsf{supp}(X, t_p, r)| \geq k$.

An itemset (X, t_p) is called *closed* in r if there is no itemset (Y, s_p) such that $(Y, s_p) \preceq (X, t_p)$ for which $\mathsf{supp}(Y, s_p, r) = \mathsf{supp}(X, t_p, r)$. For an itemset (X, t_p), we denote by $\mathsf{clo}(X, t_p)$ the unique closed itemset that extends (X, t_p) and has the same support in r as (X, t_p). Similarly, an itemset (X, t_p) is called *free* in r if there exists no itemset (Y, s_p) such that $(X, t_p) \preceq (Y, s_p)$ for which $\mathsf{supp}(Y, s_p, r) = \mathsf{supp}(X, t_p, r)$. The connection between k-frequent free and closed itemsets and k-frequent left-reduced constant CFDs forms the basis for CFDMiner, which is shown below.

Proposition 1: *For an instance r of R, there is a k-frequent left-reduced constant* CFD $\varphi = (X \to A, (t_p \parallel a))$ *such that $r \models \varphi$ iff (i) the itemset (X, t_p) is free, k-frequent and it does not contain (A, a); (ii) $\mathsf{clo}(X, t_p) \preceq (A, a)$; and (iii) (X, t_p) does not contain a smaller free set (Y, s_p) with this property, i.e., there exists no (Y, s_p) such that $(X, t_p) \preceq (Y, s_p)$, $Y \subsetneq X$, and $\mathsf{clo}(Y, s_p) \preceq (A, a)$.* □

B. CTANE: A Levelwise Algorithm

We next present CTANE, a levelwise algorithm for discovering minimal, k-frequent CFDs. It is an extension of the TANE algorithm [14] for discovering FDs. CTANE mines CFDs by traversing an attribute-set/pattern lattice $\mathcal{L}$ in a levelwise way. More precisely, the lattice $\mathcal{L}$ consists of elements of the form (X, t_p), where $X \subseteq \mathbf{attr}(R)$ and t_p is pattern tuple over X. In contrast to the itemsets in Section III-A, the patterns now consist of both constants and unnamed variables (_). Given a level ℓ in $\mathcal{L}$, we denote by L_ℓ the collection of elements (X, s_p) at this level, in which (X, s_p) has size ℓ, *i.e.,* $|X| = \ell$.

CTANE starts from L_1, *i.e.,* singleton sets (A, α) for $A \in \mathbf{attr}(R)$ and $\alpha \in \mathbf{dom}(A) \cup \{\_\}$. It then proceeds to larger attribute-set/pattern levels in $\mathcal{L}$ in a levelwise way. Similar to TANE, CTANE derives CFDs in $L_{\ell+1}$ from L_ℓ with pruning based on CFDs generated in previous levels. When the algorithm considers (X, s_p), it tests for CFDs of the form $(X \setminus \{A\} \to A, (s_p[X \setminus \{A\}] \parallel s_p[A]))$, where $A \in X$. This guarantees that only non-trivial CFDs are considered. Furthermore, CTANE maintains for each considered element (X, s_p) a set, denoted by $\mathcal{C}^+(X, s_p)$, that is used to determine whether $(X \setminus \{A\} \to A, (s_p[X \setminus \{A\}] \parallel s_p[A]))$ is minimal. The set $\mathcal{C}^+(X, s_p)$, as explained in [21], can be maintained during the levelwise traversal. Apart from testing for minimality, $\mathcal{C}^+(X, s_p)$ also provides an effective pruning strategy, making the levelwise approach feasible in practice.

C. FastCFD: A Depth First Approach

In contrast to CTANE, FastCFD discovers k-frequent minimal CFDs in a *depth-first* way. It is inspired by FastFD [15], a depth-first algorithm for discovering FDs.

Consider $X \subseteq \mathbf{attr}(R)$ and an attribute A in $\mathbf{attr}(R) \setminus X$. We denote by $\mathsf{fixlhs}(X, A, r, k)$ the set of all CFDs $\varphi = (Y \rightarrow A, t_p)$ such that $Y \subseteq X$, φ is minimal, and moreover $\sup(\varphi, r) \geqslant k$. All k-frequent CFDs in r can thus be found by computing $\bigcup_{A \in \mathbf{attr}(R)} \mathsf{fixlhs}(\mathbf{attr}(R) \setminus \{A\}, A, r, k)$. Algorithm FastCFD does exactly this: for each $A \in \mathbf{attr}(R)$, it calls a procedure FindCover that computes $\mathsf{fixlhs}(\mathbf{attr}(R) \setminus \{A\}, A, r, k)$. To compute $\mathsf{fixlhs}(\mathbf{attr}(R) \setminus \{A\}, A, r, k)$ in a depth-first way, we use difference sets as in [15].

For each subset $X \subseteq \mathbf{attr}(R) \setminus \{A\}$, FindCover maintains a list of possible k-frequent free itemsets $\mathsf{Patt}(X)$ together with its set of difference sets not covered yet. For an itemset t_p^c in $\mathsf{Patt}(X)$, we denote by $r_{t_p^c}$ the set of tuples in r that match t_p^c. For each itemset t_p^c in $\mathsf{Patt}(X)$, FindCover inspects the subsets of $\mathbf{attr}(R) \setminus \{A\}$ in a depth-first, left-to-right fashion based on an ordering of attributes on $\mathbf{attr}(R) \setminus \{A\}$ for all tuples in $r_{t_p^c}$. A candidate CFD $\varphi = (X \rightarrow A, (t_p \parallel \_))$, where t_p^c is the constant part of t_p, is produced if none of the variables (*i.e.,*'_') in $t_p[X]$ can be removed, i.e., φ is minimal in $r_{t_p^c}$. Then FindCover also ensures that the minimality conditions are checked for all subset itemsets of t_p^c in $\mathsf{Patt}(X)$ such that none of the constants in $t_p[X]$ can be removed or upgraded to '_'. This guarantees that $t_p[X]$ is the most general in r.

We implemented two approaches for computing the difference sets. The first one, called NaiveFast, is inspired by the stripped partition-based approach used by FastFD [15]. The second approach, denoted by FastCFD, relies on the availability of $\mathsf{Closed}_2(r)$, i.e., all 2-frequent closed itemsets in r. Given (X, t_p), $\mathsf{Closed}_2(r)$ can be used to infer for any two tuples in r_{t_p} on which attributes they agree. Indeed, this set of attributes is given by the attributes in those itemsets in $\mathsf{Closed}_2(r)$ that match with t_p^c (the constant part of t_p).

D. Summary of Experimental Results

From our experiments based on real-life data, i.e., Wisconsin breast cancer and chess datasets from UCI (http://archive.ics.uci.edu/ml/), and synthetic datasets generated from data scraped from the Web, we find the following: (a) CFDMiner can be multiple orders of magnitude faster than CTANE and FastCFD for constant CFD profiling. (b) CTANE usually works well when the arity of a sample relation is small and the support threshold is high, but it scales poorly when the arity of the relation increases. (c) NaiveFast and FastCFD are far more efficient than CTANE when the arity of the relation is large. (d) Our optimization technique based on closed-itemset mining is effective: FastCFD significantly outperforms NaiveFast, especially when the arity is large.

IV. Conclusions

We have developed and implemented three algorithms for discovering minimal CFDs: (1) CFDMiner for mining minimal constant CFDs, a class of CFDs important for both data cleaning and data integration; (2) CTANE for discovering general minimal CFDs based on the levelwise approach; and (3) FastCFD for discovering general minimal CFDs based on a depth-first search strategy, and a novel optimization technique via closed-itemset mining. As suggested by our experimental results, these provide a set of tools for users to choose from for different applications.

There is naturally much to be done. First, we are currently experimenting with various datasets collected from real life. Second, we are studying how to discover minimal CFDs from a dataset r when *both* its arity *and* size are large. Third, while we have employed in FastCFD techniques for mining closed itemsets, we expect that other mining techniques may also shed light on improving the performance of discovery algorithms. Fourth, we plan to explore the use of CFD inference in discovery, to eliminate CFDs that are entailed by those CFDs already found. Finally, a topic for future work is to assess various quality measures for CFDs.

References

[1] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *TODS*, vol. 33, no. 2, june 2008.

[2] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in *VLDB*, 2007.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," *TPLP*, vol. 3, no. 4-5, pp. 393–424, 2003.

[4] J. Chomicki and J. Marcinkowski, "Minimal-change integrity maintenance using tuple deletions," *Information and Computation*, vol. 197, no. 1-2, pp. 90–121, 2005.

[5] J. Wijsen, "Database repairing using updates," *TODS*, vol. 30, no. 3, pp. 722–768, 2005.

[6] C. Batini and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[7] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches." *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.

[8] Gartner, "Forecast: Data quality tools, worldwide, 2006-2011," 2007.

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

[10] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies," in *VLDB*, 2008.

[11] P. Brown and P. J. Haas, "BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data," in *VLDB*, 2003.

[12] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga, "Automatic discovery of correlations and soft functional dependencies," in *SIGMOD*, 2004.

[13] H. Mannila and K.-J. Räihä, "Dependency inference," in *VLDB*, 1987.

[14] Y. Huhtala, J. Kärkk ainen, P. Porkka, and H. Toivonen, "TANE: An efficient algorithm for discovering functional and approximate dependencies," *Comput. J.*, vol. 42, no. 2, pp. 100–111, 1999.

[15] C. M. Wyss, C. Giannella, and E. L. Robertson, "FastFDs: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances - extended abstract," in *DaWak*, 2001.

[16] P. A. Flach and I. Savnik, "Database dependency discovery: A machine learning approach," *AI Commun.*, vol. 12, no. 3, pp. 139–160, 1999.

[17] S. Lopes, J.-M. Petit, and L. Lakhal, "Efficient discovery of functional dependencies and armstrong relations," in *EDBT*, 2000.

[18] T. Calders, R. T. Ng, and J. Wijsen, "Searching for dependencies at multiple abstraction levels," *TODS*, vol. 27, no. 3, pp. 229–260, 2003.

[19] R. S. King and J. J. Legendre, "Discovery of functional and approximate functional dependencies in relational databases," *JAMDS*, vol. 7, no. 1, pp. 49–59, 2003.

[20] F. Chiang and R. Miller, "Discovering data quality rules," in *VLDB*, 2008.

[21] Full version, *http://www.lfcs.inf. ed.ac.uk/research/ database/publications/profiling.pdf*.

[22] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," in *Advances in Knowledge Discovery and Data Mining*, 1996.

[23] M. J. Zaki, "Mining non-redundant association rules," *Data Min. Knowl. Discov.*, vol. 9, no. 3, pp. 223–248, 2004.

[24] J. Li, G. Liu, and L. Wong, "Mining statistically important equivalence classes and delta-discriminative emerging patterns," in *KDD*, 2007.