Extending Conditional Dependencies with Built-in Predicates

Citation for published version:
Ma, S, Duan, L, Fan, W, Hu, C & Chen, W 2015, 'Extending Conditional Dependencies with Built-in Predicates', *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3274 - 3288. https://doi.org/10.1109/TKDE.2015.2451632

Digital Object Identifier (DOI):
10.1109/TKDE.2015.2451632

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
IEEE Transactions on Knowledge and Data Engineering

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

Extending Conditional Dependencies with Built-in Predicates

Shuai Ma, Liang Duan, Wenfei Fan, Chunming Hu, and Wenguang Chen

Abstract—This paper proposes a natural extension of conditional functional dependencies (CFDs [22]) and conditional inclusion dependencies (CINDs [30]), denoted by CFD^{p}s and CIND^{p}s, respectively, by specifying patterns of data values with $\neq, <, \leq, >$ and $\geq$ predicates. As data quality rules, CFD^{p}s and CIND^{p}s are able to capture errors that commonly arise in practice but cannot be detected by CFDs and CINDs. We establish two sets of results for central technical problems associated with CFD^{p}s and CIND^{p}s. (a) One concerns the satisfiability and implication problems for CFD^{p}s and CIND^{p}s, taken separately or together. These are important for, *e.g.*, deciding whether data quality rules are dirty themselves, and for removing redundant rules. We show that despite the increased expressive power, the static analyses of CFD^{p}s and CIND^{p}s retain the same complexity as their CFDs and CINDs counterparts. (b) The other concerns validation of CFD^{p}s and CIND^{p}s. We show that given a set Σ of CFD^{p}s and CIND^{p}s on a database D, a set of SQL queries can be automatically generated that, when evaluated against D, return all tuples in D that violate some dependencies in Σ. We also experimentally verified the efficiency and effectiveness of our SQL based error detection techniques, using real-life data. This provides commercial DBMS with an immediate capability to detect errors based on CFD^{p}s and CIND^{p}s.

Index Terms—Conditional dependencies, built-in predicates, functional dependencies, inclusion dependencies, data quality

1 Introduction

Extensions of traditional functional dependencies (FDs) and inclusion dependencies (INDs), known as *conditional functional dependencies* (CFDs [22]) and *conditional inclusion dependencies* (CINDs [30]), respectively, have recently been proposed for improving data quality. These extensions enforce patterns of semantically related data values, and detect errors as violations of the dependencies. It has been shown that conditional dependencies are able to capture more inconsistencies than FDs and INDs [17], [21], [30].

Conditional dependencies specify constant patterns in terms of equality ($=$). In practice, however, the semantics of data often need to be specified with other predicates such as $\neq, <, \leq, >$ and $\geq$, as illustrated by the following example.

Example 1: An online store maintains a database of two relations: (a) item for items sold by the store, and (b) tax for the sale tax rates for the items, except artwork, in various states. The relations are specified by the following schemas:

item (id: string, name: string, type: string, price: float,
  shipping: float, sale: bool, state: string)
tax (state: string, rate: float)

where each item is specified by its id, name, type (*e.g.*, book, CD), price, shipping fee, the state to which it is shipped, and whether it is on sale. A tax tuple specifies the sale tax rate in a state. An instance D_0 of item and tax is shown in Fig. 1.

One wants to specify dependencies on the relations as data quality rules to detect errors in the data, such that inconsistencies emerge as violations of the dependencies. Traditional dependencies (FDs, INDs; see, *e.g.*, [3]) and conditional dependencies (CFDs, CINDs [22], [30]) on the data include the following:

cfd$_1$: item (id $\rightarrow$ name, type, price, shipping, sale)
cfd$_2$: tax (state $\rightarrow$ rate)
cfd$_3$: item (sale = 'T' $\rightarrow$ shipping = 0)

These are CFDs: (a) cfd$_1$ assures that the id of an item uniquely determines its name, type, price, shipping and sale; (b) cfd$_2$ states that state is a key for tax, *i.e.*, for each state there is a unique sale tax rate; and (c) cfd$_3$ ensures that for any item tuple t, if t[sale] = 'T' then t[shipping] must be 0; *i.e.*, free shipping is provided for items on sale. Here cfd$_3$ is specified in terms of patterns of semantically related data values, namely, sale = 'T' and shipping = 0. It is to hold only on item tuples that match the pattern sale = 'T'. In contrast, cfd$_1$ and cfd$_2$ are traditional FDs without constant patterns, a special case of CFDs. One can verify that no sensible INDs or CINDs can be defined across item and tax.

Note that D_0 of Fig. 1 satisfies cfd$_1$, cfd$_2$ and cfd$_3$. That is, when these dependencies are used as data quality rules, no errors are found in D_0.

In practice, the shipment fee of an item is typically determined by the price of the item. Moreover, when an item is on sale, the price of the item is often in a certain range. Furthermore, for any item sold by the store to a customer in a state, if the item is *not* artwork, then one expects to find the sale tax rate in the state from the tax table. These semantic relations cannot be expressed as CFDs of [22] or CINDs of [30], but can be expressed as the following dependencies:

- *S. Ma, L. Duan and C. Hu are with the SKLSDE lab, School of Computer Science and Engineering, Beihang University, China. E-mail: {mashuai, duanl, hucm}@act.buaa.edu.cn.*
- *W. Fan is with the RCBD center, Beihang University, China and the School of Informatics, Edinburgh University, UK. E-mail: wenfei@inf.ed.ac.uk.*
- *W. Chen is with the Department of Information Management, Peking University, China. E-mail: chenwg@pku.edu.cn.*

Manuscript received XXX, 2014; revised XXX, 2014.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

	id	name	type	price	shipping	sale	state
t_1:	b1	Harry Potter	book	25.99	0	T	WA
t_2:	c1	Snow White	CD	9.99	2	F	NY
t_3:	b2	Catch-22	book	34.99	20	F	DL
t_4:	a1	Sunflowers	art	5m	500	F	DL

(a) An item relation

	state	rate
t_5:	PA	6
t_6:	NY	4
t_7:	DL	0
t_8:	NJ	3.5

(b) A tax rate relation

Figure 1. Example instance D_0 of item and tax

pfd$_1$: item (sale = 'F' **and** price $\leq 20 \rightarrow$ shipping = 3)
pfd$_2$: item (sale = 'F' **and** price > 20 **and** price ≤ 40 $\rightarrow$ shipping = 6)
pfd$_3$: item (sale = 'F' **and** price $> 40 \rightarrow$ shipping = 10)
pfd$_4$: item (sale = 'T' $\rightarrow$ price ≥ 2.99 **and** price < 9.99)
pind$_1$: item (state; type $\neq$ 'art') $\subseteq$ tax (state; nil)

Here pfd$_2$ states that for any item tuple, if it is not on sale and its price is in the range $(20, 40]$, then its shipment fee must be 6; similarly for pfd$_1$ and pfd$_3$. These dependencies extend CFDs [22] by specifying patterns of semantically related data values in terms of predicates $<$, $\leq$, $>$ and $\geq$. Similarly, pfd$_4$ assures that for any item tuple, if it is on sale, then its price must be in the range $[2.99, 9.99)$. Finally, pind$_1$ extends CINDs [30] by specifying patterns with $\neq$: for any item tuple t, if t[type] is *not* artwork, then there must exist a tax tuple t' such that t[state] $= t'$[state], *i.e.*, the sale tax of the item can be found from the tax relation.

Using dependencies pfd$_1$–pfd$_4$ and pind$_1$ as data quality rules, we find that D_0 of Fig. 1 is *not* clean. Indeed, (a) t_2 violates pfd$_1$: its price is less than 20, but its shipping fee is 2 rather than 3; similarly, t_3 violates pfd$_2$, and t_4 violates pfd$_3$. (b) Tuple t_1 violates pfd$_4$: it is on sale but its price is not in the range $[2.99, 9.99)$. (c) The database D_0 also violates pind$_1$: t_1 is not artwork, but its state cannot find a match in the tax relation, *i.e.*, no tax rate for WA is found in D_0. □

None of pfd$_1$–pfd$_4$ and pind$_1$ can be expressed as FDs or INDs [3], which do not allows constants, or as CFDs [22] or CINDs [30], which specify patterns with equality $(=)$ only. While there have been extensions of CFDs [10], [13], [28], none of these allows dependencies to be specified with patterns on data values in terms of built-in predicates $\neq, <, \leq, >$ or $\geq$. To the best of our knowledge, the earlier conference version [12] of this paper is the first to study these constraints.

These highlight the need for extending CFDs and CINDs to capture errors commonly found in real-life data. While one can consider arbitrary extensions, it is necessary to strike a balance between their expressive power and their complexity. In particular, we want to be able to reason about data quality rules expressed as extended CFDs and CINDs. Furthermore, we want to have effective algorithms to detect inconsistencies based on these extensions.

Contributions & Roadmap. To this end we introduce an extension of CFDs and CINDs, investigate the static analyses of these constraints, and develop effective SQL-based techniques for detecting errors based on these constraints.

(1) We propose two classes of dependencies, denoted by CFD^{p}s and CIND^{p}s, which respectively extend CFDs and CINDs by supporting $\neq$, $<$, $\leq$, $>$, $\geq$ predicates (Sections 2 and 3). For example, all the dependencies we have encountered so far can be expressed as CFD^{p}s or CIND^{p}s. These dependencies are capable of capturing errors in real-world data that cannot be detected by CFDs or CINDs.

(2) We establish the complexity bounds for the satisfiability and implication problems for CFD^{p}s and CIND^{p}s, taken separately or together (Section 4). The satisfiability problem is to determine whether a set Σ of dependencies has a nonempty model, *i.e.*, whether the rules in Σ are consistent themselves. The implication problem is to decide whether a set Σ of dependencies entails another dependency φ, *i.e.*, whether the rule φ is redundant in the presence of the rules in Σ. These are the central technical problems associated with any dependency language.

We show that despite the increased expressive power, CFD^{p}s and CIND^{p}s do not increase the complexity for reasoning about them. In particular, we show that the satisfiability and implication problems remain (a) NP-complete and coNP-complete for CFD^{p}s, respectively, (b) in $O(1)$-time (constant-time) and EXPTIME-complete for CIND^{p}s, respectively, and (c) are undecidable when CFD^{p}s and CIND^{p}s are taken together. These are *the same as* their CFDs and CINDs counterparts [30]. In contrast, data with linearly ordered domains often makes our lives harder [35].

(3) We provide SQL-based techniques to detect errors based on CFD^{p}s and CIND^{p}s (Section 5). Given a set Σ of CFD^{p}s and CIND^{p}s on a database D, we automatically generate a set of SQL queries that, when evaluated on D, find all tuples in D that violate some dependencies in Σ. Further, the SQL queries are independent of the size and cardinality of Σ. These provide the capability of detecting errors in a single relation (CFD^{p}s) and across different relations (CIND^{p}s) within the immediate reach of commercial DBMS.

(4) Using real-life data (HOSP and DBLP), we finally conduct an extensive experimental study (Section 6). We show that (a) the running time of CFD^{p}s and CIND^{p}s is comparable to their CFDs and CINDs counterparts, which is consistent with the static analyses in Section 4, and (b) CFD^{p}s and CIND^{p}s are able to capture more errors than their CFDs and CINDs counterparts (22% on HOSP and 75% on DBLP), due to the increased expressive power.

Related work. This paper is an extension of our earlier work [12] by adding (a) the proofs for the complexity bounds for the satisfiability and implication analyses of CFD^{p}s and CIND^{p}s, separately and taken together (Section 4), and (b) an extensive experimental study of CFD^{p}s and CIND^{p}s (Section 6), which was not investigated in [12].

Recently, data dependencies have generated renewed interests for improving data quality [5], [10], [14], [15], [22], [28], [30], [33], [36]. Constraint-based data cleaning was introduced in [4], which proposed to use dependencies, *e.g.*, FDs, INDs and denial constraints, to detect and repair errors in real-life data (see, *e.g.*, [3], [15], [33] for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

(1) φ_1 = tax (state $\to$ rate, T_1)

T_1:

state	rate
_	_

(2) φ_2 = item (sale $\to$ shipping, T_2)

T_2:

sale	shipping
= T	= 0

(3) φ_3 = item (sale, price $\to$ shipping, T_3)

T_3:

sale	price	shipping
$=$ F	> 20	$= 6$
$=$ F	≤ 40	$= 6$

(4) CFD^p φ_4 = item (sale $\to$ price, T_4)

T_4:

sale	price
$=$ T	≥ 2.99
$=$ T	< 9.99

Figure 2. Example CFD^{p}s

details). Data dependencies have been studied for relational databases since the introduction of FDs by Codd [16] in 1972 (see, *e.g.*, [3] for details), and the theory of INDs was established in [11], which developed a sound and complete inference system and the PSPACE-completeness for the implication analysis of INDs. As an extension of traditional FDs, CFDs were developed in [22], for improving the quality of data. It was shown in [22] that the satisfiability and implication problems for CFDs are NP-complete and coNP-complete, respectively. Along the same lines, CINDs [30] were proposed to extend INDs, and it was shown [30] that the satisfiability and implication problems for CINDs are in constant time and EXPTIME-complete, respectively. SQL techniques were developed in [22] to detect errors by using CFDs, but have not been studied for CINDs. This work extends the static analyses of conditional dependencies of [22], [30], and has established several new complexity results, notably in the absence of finite-domain attributes (*e.g.*, Theorems 2, 8 and Proposition 6). In addition, it is the first work to develop SQL techniques for checking violations of CINDs and violations of CFD^{p}s and CIND^{p}s taken together.

Extensions of CFDs have been proposed to support disjunction and negation [10], cardinality constraints and synonym rules [13], and to specify patterns in terms of value ranges [28]. While CFD^{p}s are more powerful than the extension of [28], they cannot express disjunctions [10], cardinality constraints and synonym rules [13]. To our knowledge no extensions of CINDs have been studied. This work is the first full treatment of extensions of CFDs and CINDs by incorporating built-in predicates ($\neq, <, \leq, >, \geq$), from static analyses to error detection.

Methods have been developed for discovering CFDs [14], [28], CFD^{p}s [36] and CINDs [5] and for repairing data based on either CFDs [17], traditional FDs and INDs taken together [8], CFDs and CINDs taken together [19], denial constraints [7], aggregate constraints [25], matching dependencies [20], matching dependencies and CFDs [24], or editing rules and master data [23]. We defer the treatment of these topics for CFD^{p}s and CIND^{p}s to future work.

A variety of extensions of FDs and INDs have been studied for specifying constraint databases and constraint logic programs [6], [9], [27], [31], [32]. While the languages of [6], [27], [31] cannot express CFD^{p}s, constraint-generating dependencies (CGDs) of [6] and constrained tuple-generating dependencies (CTGDs) of [32] can express CFD^{p}s, and CTGDs can also express CIND^{p}s. The increased expressive power of CTGDs comes at the price of a higher complexity: both their satisfiability and implication problems are undecidable. Built-in predicates and arbitrary constraints are supported by CGDs, for which it is not clear whether effective SQL queries can be developed to detect errors. It is worth mentioning that Theorems 2 and 6 of this work provide lower bounds for the consistency and implication analyses of CGDs, by using patterns with built-in predicates only.

Observe that constraints specifying semantics with orderings have long been recognized, such as order dependencies [27] supporting the comparison of attributes with $=, <, \leq, >, \geq$, matching dependencies [20] and differential dependencies [34] that support the comparison of attributes with $=, \neq, <, \leq, >, \geq$ for record matching. However, different from CFDs and CFD^{p}s, these constraints do not specify conditions on those tuples such that the embedded FDs hold. Further, it is also possible that other existing constraints could be improved by incorporating these built-in predicates, such as metric functional dependencies [29].

2 Extending CFDs with Predicates

We now define *conditional functional dependencies with predicates*, denoted by CFD^{p}s, by extending CFDs [22] with built-in predicates ($\neq$, $<$, $\leq$, $>$, $\geq$) in addition to equality ($=$).

Consider a relational schema R defined over a finite set of attributes, denoted by $\mathsf{attr}(R)$. For each attribute $A \in \mathsf{attr}(R)$, its domain is specified in R, denoted as $\mathsf{dom}(A)$, which is either finite (*e.g.*, bool) or infinite (*e.g.*, string). We assume *w.l.o.g.* that a domain on which $<$, $\leq$, $>$ or $\geq$ is defined is totally ordered.

Syntax. A CFD^p φ on R is a pair $R(X \to Y,\ T_p)$, where (1) X, Y are sets of attributes in $\mathsf{attr}(R)$; (2) $X \to Y$ is a standard FD, referred to as the FD *embedded in* φ; and (3) T_p is a tableau with attributes in X and Y, referred to as the *pattern tableau* of φ, where for each A in $X \cup Y$ and each tuple $t_p \in T_p$, $t_p[A]$ is either an unnamed variable '_' that draws values from $\mathsf{dom}(A)$, or 'op a', where op is one of $\{=, \neq, <, \leq, >, \geq\}$, and '$a$' is a constant in $\mathsf{dom}(A)$.

If attribute A occurs in both X and Y, we use A_L and A_R to indicate the occurrence of A in X and Y, respectively, and we separate the X and Y attributes in a pattern tuple with '$\|$'. We simply write φ as $(X \to Y,\ T_p)$ when R is clear from the context, and denote X as $\mathsf{LHS}(\varphi)$ and Y as $\mathsf{RHS}(\varphi)$, respectively.

Example 2: The dependencies cfd_1–cfd_3 and pfd_1–pfd_4 that we have seen in Example 1 can all be expressed as CFD^{p}s. Some of these CFD^{p}s are illustrated in Fig. 2, in which φ_1 is for FD cfd_2, φ_2 is for CFD cfd_3, φ_3 is for pfd_2, and φ_4 is for pfd_4, respectively. □

Semantics. Consider CFD^p $\varphi = R(X \to Y, T_p)$, where $T_p = \{t_{p_1}, \ldots, t_{p_k}\}$.

A data tuple t of R is said to *match* $\mathsf{LHS}(\varphi)$, denoted by $t[X] \asymp T_p[X]$, if *for each tuple* t_{p_i} *(*$i \in [1, k]$*) in* T_p and *each*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

attribute A in X, either (a) $t_{p_i}[A]$ is the wildcard '_' (which matches any value in $\mathsf{dom}(A)$), or (b) $t[A]$ op a if $t_{p_i}[A]$ is 'op a', where the operator op $(=, \neq, <, \leq, >$ or $\geq)$ is interpreted by its standard semantics. Similarly, the notion that t *matches* $\mathsf{RHS}(\varphi)$ is defined, denoted by $t[Y] \asymp T_p[Y]$.

Intuitively, each pattern tuple t_{p_i} $(i \in [1, k])$ specifies a condition via $t_{p_i}[X]$, and $t[X] \asymp T_p[X]$ if $t[X]$ satisfies the *conjunction* of all these conditions. Similarly, $t[Y] \asymp T_p[Y]$ if $t[Y]$ matches all the patterns specified by $t_{p_i}[Y]$ for all pattern tuples t_{p_i} in T_p.

An instance I of R *satisfies* the CFD^p φ, denoted by $I \models \varphi$, if for *each pair* of tuples t_1, t_2 in I, if $t_1[X] = t_2[X] \asymp T_p[X]$, then $t_1[Y] = t_2[Y] \asymp T_p[Y]$. That is, if $t_1[X]$ and $t_2[X]$ are equal and in addition, they both match the pattern tableau $T_p[X]$, then $t_1[Y]$ and $t_2[Y]$ must also be equal to each other and they both match the pattern tableau $T_p[Y]$.

Observe that φ is imposed only on the subset of tuples in I that match $\mathsf{LHS}(\varphi)$, rather than on the entire I. For all tuples t_1, t_2 in this subset, if $t_1[X] = t_2[X]$, then (a) $t_1[Y] = t_2[Y]$, *i.e.,* the semantics of the embedded FDs is enforced; and (b) $t_1[Y] \asymp T_p[Y]$, which assures that the *constants* in $t_1[Y]$ match the *constants* in $t_{p_i}[Y]$ for all t_{p_i} in T_p. Note that here tuples t_1 and t_2 can be the same.

An instance I of R satisfies a set Σ of CFD^{p}s, denoted by $I \models \Sigma$, if $I \models \varphi$ for *each* CFD^p φ in Σ.

Example 3: The instance D_0 of Fig. 1 satisfies φ_1 and φ_2 of Fig. 2, but neither φ_3 nor φ_4. Indeed, tuple t_3 violates (*i.e.,* does not satisfy) φ_3, since $t_3[\mathsf{sale}]$ = 'F' *and* $20 < t_3[\mathsf{price}] \leq 40$, but $t_3[\mathsf{shipping}]$ is 20 instead of 6. Note that t_3 matches $\mathsf{LHS}(\varphi_3)$ since it satisfies the condition specified by the *conjunction* of the pattern tuples in T_3. Similarly, t_1 violates φ_4, since $t_1[\mathsf{sale}]$ = 'T' but $t_1[\mathsf{price}] > 9.99$. Observe that while it takes two tuples to violate a standard FD, a single tuple may violate a CFD^p. □

Special cases. (1) A standard FD $X \rightarrow Y$ [3] can be expressed as a CFD $(X \rightarrow Y,\ T_p)$ in which T_p contains a single tuple consisting of '_' only, without constants. (2) A CFD $(X \rightarrow Y,\ T_p)$ [22] with $T_p = \{t_{p1}, \ldots, t_{pk}\}$ can be expressed as a set $\{\varphi_1, \ldots, \varphi_k\}$ of CFD^{p}s such that for each $i \in [1, k]$, $\varphi_i = (X \rightarrow Y,\ T_{p_i})$, where T_{p_i} contains the pattern tuple t_{p_i} of T_p only, defined with equality $(=)$ only. For example, φ_1 and φ_2 in Fig. 2 are CFD^{p}s representing FD cfd_2 and CFD cfd_3 in Example 1, respectively. Note that all data quality rules in [14], [28] can be expressed as CFD^{p}s.

3 Extending CINDs with Predicates

Similar to CFD^{p}s, we define *conditional inclusion dependencies with predicates*, denoted by CIND^{p}s, by extending CINDs [30] with built-in predicates $(\neq, <, \leq, >, \geq)$ in addition to equality $(=)$. Consider two relational schemas R_1 and R_2.

Syntax. A CIND^p ψ is a pair $(R_1[X;\ X_p] \subseteq R_2[Y;\ Y_p],\ T_p)$, where (1) X, X_p and Y, Y_p are lists of attributes in $\mathsf{attr}(R_1)$ and $\mathsf{attr}(R_2)$, respectively; (2) $R_1[X] \subseteq R_2[Y]$ is a standard IND, referred to as the IND *embedded* in ψ; and (3) T_p is a tableau, called the *pattern tableau* of ψ defined over attributes $X_p \cup Y_p$, and for each A in X_p or Y_p and each pattern tuple $t_p \in T_p$, $t_p[A]$ is either an unnamed variable '_' that draws values from $\mathsf{dom}(A)$, or 'op a', where op is one of $=, \neq, <, \leq, >, \geq$ and 'a' is a constant in $\mathsf{dom}(A)$.

We denote $X \cup X_p$ as $\mathsf{LHS}(\psi)$, $Y \cup Y_p$ as $\mathsf{RHS}(\psi)$, and separate the X_p and Y_p attributes in a pattern tuple with '$\|$'. We also use nil to denote an *empty* list.

Example 4: Figure 3 shows two example CIND^{p}s: ψ_1 expresses the pind_1 in Example 1, and ψ_2 refines ψ_1 by stating that for any item tuple t_1, if its type is not art and its state is DL, then there must be a tax tuple t_2 such that its state is DL and rate is 0, *i.e.,* ψ_2 assures that the sale tax rate in Delaware is 0. □

Semantics. Consider CIND^p $\psi = (R_1[X;\ X_p] \subseteq R_2[Y;\ Y_p],\ T_p)$. An instance (I_1, I_2) of (R_1, R_2) *satisfies* the CIND^p ψ, denoted by $(I_1, I_2) \models \psi$, iff for *each* tuple $t_1 \in I_1$, if $t_1[X_p] \asymp T_p[X_p]$, then there *exists* a tuple $t_2 \in I_2$ such that $t_1[X] = t_2[Y]$ and $t_2[Y_p] \asymp T_p[Y_p]$.

That is, if $t_1[X_p]$ matches the pattern tableau $T_p[X_p]$, then ψ assures the existence of t_2 such that (1) $t_1[X] = t_2[Y]$ as needed by the standard IND embedded in ψ; and, moreover, (2) $t_2[Y_p]$ must match the pattern tableau $T_p[Y_p]$. In other words, ψ is "conditional" since its embedded IND is applied only to the subset of tuples in I_1 that match $T_p[X_p]$, and $T_p[Y_p]$ is enforced on the tuples in I_2 that match those tuples in I_1. As remarked in Section 2, the pattern tableau T_p specifies the *conjunction* of all the pattern tuples in T_p.

Example 5: The instance D_0 of item and tax in Fig. 1 violates CIND^p ψ_1. Indeed, tuple t_1 in item *matches* $\mathsf{LHS}(\psi_1)$ since $t_1[\mathsf{type}] \neq$ 'art', but there is no tuple t in tax such that $t[\mathsf{state}]$ = $t_1[\mathsf{state}]$ = 'WA'. In contrast, D_0 satisfies ψ_2. □

We say that a database D satisfies a set Σ of CIND^{p}s, denoted by $D \models \Sigma$, if $D \models \psi$ for each $\psi \in \Sigma$.

Safe CIND^{p}s. We say a CIND^p $(R_1[X;\ X_p] \subseteq R_2[Y; Y_p],\ T_p)$ is *unsafe* if there exist pattern tuples t_p, t'_p in T_p such that either (a) there exists $B \in Y_p$, such that $t_p[B]$ and $t'_p[B]$ are not satisfiable when taken together, or (b) there exist $C \in Y$, $A \in X$ such that A corresponds to C in the embedded IND and $t_p[C]$ and $t'_p[A]$ are not satisfiable when taken together; *e.g.,* $t_p[\mathsf{price}] = 9.99$ and $t'_p[\mathsf{price}] \geq 19.99$.

Obviously unsafe CIND^{p}s do not make sense: no nonempty databases satisfy unsafe CIND^{p}s. It takes $O(|T_p|^2)$ time in the size $|T_p|$ of T_p to decide whether a CIND^p is unsafe. Thus in the sequel we consider safe CIND^p only.

Special cases. (1) A standard IND $(R_1[X] \subseteq R_2[Y])$ can be expressed as a CIND^p $(R_1[X;\ nil] \subseteq R_2[Y;\ nil],\ T_p)$ such that T_p is simply a *empty* set. (2) A CIND $(R_1[X;\ X_p] \subseteq R_2[Y;\ Y_p],\ T_p)$ with $T_p = \{t_{p1}, \ldots, t_{pk}\}$ can be expressed as a set $\{\psi_1, \ldots, \psi_k\}$ of CIND^{p}s, where for each $i \in [1, k]$, $\psi_i = (R_1[X;\ X_p] \subseteq R_2[Y;\ Y_p],\ T_{pi})$ such that T_{pi} consists of the pattern tuple t_{pi} of T_p, defined with equality $(=)$ only.

4 Reasoning about CFD^{p}s and CIND^{p}s

The satisfiability and implication problems are the two classical questions associated with any dependency languages [3], [22], [30]. In this section we investigate these problems for CFD^{p}s and CIND^{p}s, separately and taken together.

4.1 Satisfiability Analyses

The satisfiability problem is to determine, given a set Σ of constraints, whether there exists a *nonempty* database that satisfies Σ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

(1) ψ_1 = (item [state; type] $\subseteq$ tax [state; *nil*], T_1),

(2) ψ_2 = (item [state; type, state] $\subseteq$ tax [state; rate], T_2)

T_1:

type	nil
$\neq$ art	

T_2:

type	state	rate
$\neq$ art	$=$ DL	$= 0$

Figure 3. Example CIND^{p}s

The satisfiability analysis of conditional dependencies is not only of theoretical interest, but is also important in practice. Indeed, when CFD^{p}s and CIND^{p}s are used as data quality rules, this analysis helps one check whether the rules make sense themselves. The need for this is particularly evident when the rules are manually designed or discovered from various datasets [5], [14], [28].

Satisfiability analysis of CFD^{p}s. Given any FDs, one does not need to worry about their satisfiability as any set of FDs is always satisfiable. However, as observed in [22], for a set Σ of CFDs on a relational schema R, there may not exist a *nonempty* instance I of R such that $I \models \Sigma$. As CFDs are a special case of CFD^{p}s, the same problem exists when it comes to CFD^{p}s.

Example 6: Consider a CFD^p $\varphi = (R : A \rightarrow B,\ T_p)$ such that $T_p = \{(\_ \,\|\!= a), (\_ \,\|\!\neq a)\}$. There is no *nonempty* instance I of R that satisfies φ. Indeed, for any R tuple t, φ requires that both $t[B] = a$ and $t[B] \neq a$, which is impossible. □

This problem is already NP-complete for CFDs [22]. Below we show that it remains the same complexity for CFD^{p}s despite their increased expressive power.

Proposition 1: *The satisfiability problem for* CFD^p*s is* NP-*complete.* □

Proof: The lower bound follows from the NP-hardness of their CFDs counterparts [22], since CFDs are a special case of CFD^{p}s. The upper bound is verified by presenting an NP algorithm that, given a set Σ of CFD^{p}s defined on a relational schema R, determines whether Σ is satisfiable.

We next present an NP algorithm that, given a set Σ of CFD^{p}s defined on a relational schema R, determines whether Σ is satisfiable or not. The satisfiability problem has the following small model property: If there is a nonempty R instance I such that $I \models \Sigma$, then for any tuple $t \in I$, instance I_t =$\{t\}$ satisfies Σ. Thus it suffices to consider single-tuple instances $I = \{t\}$ for deciding whether Σ is satisfiable.

Assume *w.l.o.g.* that the attributes attr(R) = $\{A_1, \ldots, A_m\}$ and the total number of pattern tuples in all pattern tableaux T_p of CFD^{p}s in Σ is h. For each $i \in [1, m]$, define the active domain of A_i to be a set $\mathsf{adom}(A_i) = C_0 \cup C_1$, where (1) C_0 consists of all constants in $T_p[A_i]$ of all pattern tableaux T_p in Σ, and if C_0 is empty, we further let $C_0 = \{a_1, a_2\}$, where $a_1, a_2 \in \mathsf{dom}(A_i)$ and $a_1 \neq a_2$, and (2) C_1 contains the set of constants for the attributes whose domains have total orders, *i.e.,* involved with predicates $\neq, <, \leq, >$ or $\geq$:

(1) Arrange all constants in C_0 in the increasing order, and assume the resulting $C_0 = \{a_1, \ldots, a_k\}$ $(k \geq 1)$;
(2) Add a constant $b_{01} \in \mathsf{dom}(A_i)$ to C_1 such that $b_{01} < a_1$ if there exists one; And also add another constant $b_{02} \in \mathsf{dom}(A_i)$ to C_1 such that $b_{02} < a_1$ and $b_{02} \neq b_{01}$ if there exists one;
(3) Similarly, for each $j \in [1, k-1]$, add a constant $b_{j1} \in \mathsf{dom}(A_i)$ to C_1 such that $a_j < b_{j1} < a_{j+1}$ if there exists one; And also add another constant $b_{j2} \in \mathsf{dom}(A_i)$ to C_1 such that $a_j < b_{j2} < a_{j+1}$ and $b_{j2} \neq b_{j1}$ if there exists one;
(4) Finally, add a constant $b_{k1} \in \mathsf{dom}(A_i)$ to C_1 such that $b_{k1} > a_k$ if there exists one; And also add another constant $b_{k2} \in \mathsf{dom}(A_i)$ to C_1 such that $b_{k2} > a_k$ and $b_{k2} \neq b_{k1}$ if there exists one.

Moveover, the number of elements in $\mathsf{adom}(A_i)$ is at most $3 * h + 2$. Then one can easily verify that Σ is satisfiable iff there exists a mapping ρ from $t[A_i]$ to $\mathsf{adom}(A_i)$ $(i \in [1, m])$ such that $I = \{(\rho(t[A_1]), \ldots, \rho(t[A_m]))\}$ and $I \models \Sigma$.

We now give an NP algorithm as follows: (1) Guess an instance, which contains a single tuple t of R such that $t[A_i] \in \mathsf{adom} A_i$ for each $i \in [1, m]$. (2) Check whether $I \models \Sigma$. If so the algorithm returns 'yes', and otherwise it repeats steps (1) and (2). Obviously step (2) can be done in PTIME in the size of Σ. Hence the algorithm is in NP, and so is the problem. □

It is known [22] that the satisfiability problem for CFDs is in PTIME when the CFDs considered are defined over attributes that have an infinite domain, *i.e.,* in the absence of finite domain attributes. However, this is no longer the case for CFD^{p}s. This tells us that the increased expressive power of CFD^{p}s does take a toll in this special case. It should be remarked that while the proof of Proposition 1 is an extension of its counterpart in [22], the result below is new.

Theorem 2: *In the absence of finite domain attributes, the satisfiability problem for* CFD^p*s remains* NP-*complete.* □

Proof: The problem is in NP by Proposition 1. Its NP-hardness is shown by reduction from the 3SAT problem, which is NP-complete (cf. [26]).

We next show the reduction from the 3SAT problem. Consider an instance $\phi = C_1 \wedge \cdots \wedge C_n$ of 3SAT, where all the variables in ϕ are $x_1, \ldots, x_m$, C_j is of the form $y_{j_1} \vee y_{j_2} \vee y_{j_3}$ such that for each $i \in [1, 3]$, y_{j_i} is either $x_{p_{ji}}$ or $\overline{x_{p_{ji}}}$ for $p_{ji} \in [1, m]$. Given ϕ, we construct a relational schema R and a set Σ of CFD^{p}s defined on R such that ϕ is satisfiable iff Σ is satisfiable.

(1) We first define the relational schema $R(X_1, \ldots, X_m, C_1, \ldots, C_n, Z)$, where all attributes share a common infinite domain dom that contains constant a. Intuitively, for each R tuple t, $t[X_1, \ldots, X_m]$ specifies a truth assignment ξ for variables $x_1, \ldots, x_m$ of ϕ, and $t[C_i]$ $(i \in [1, n])$ and $t[Z]$ are the truth values of clause C_i and sentence ϕ *w.r.t.* the assignment ξ, respectively.

(2) We then construct the set CFD^{p}s $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \ldots \cup \Sigma_n \cup \Sigma_{n+1}$, defined as follows.

(a) Σ_0 contains $n+1$ CFD^{p}s, which intuitively encode the relationships of the truth values between the clauses

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

$C_1, \ldots, C_n$ and sentence ϕ.
For each clause C_i ($i \in [1, n]$), we add to Σ_0 a CFD^p $\varphi_i = (C_1, \ldots, C_n \rightarrow Z, T_{pi})$, in which $T_{pi} = \{t_{pi}\}$ such that $t_{pi}[C_i, Z] = (\neq a \parallel \neq a)$ and $t_{pi}[C_j]$ = '_' for any $j \neq i$ and $j \in [1, n]$. We also add to Σ_0 a CFD^p $\varphi_0 = (C_1, \ldots, C_n \rightarrow Z, T_{p0})$, where $T_{p0} = \{(= a, \ldots, = a \parallel = a)\}$. Intuitively, we use $\neq a$ and $= a$ to represent *false* and *true*, respectively.

(b) For $i \in [1, n]$, Σ_i contains 8 CFD^{p}s, which intuitively encode the relationships of the truth values between the clause C_i and its three variables.
For clause $C_i = x_{j_1} \vee \overline{x_{j_2}} \vee x_{j_3}$ of ϕ with $1 \leq j_1, j_2, j_3 \leq m$, we define CFD^{p}s (a) $\varphi_{i,0} = (X_{j_1}, X_{j_2}, X_{j_3} \rightarrow C_i, T_{pi,0})$ with $T_{pi,0} = \{(< a, < a, < a \parallel = a)\}$, (b) $\varphi_{i,1} = (X_{j_1}, X_{j_2}, X_{j_3} \rightarrow C_i, T_{pi,2})$ with $T_{pi,1} = \{(< a, < a, \geq a \parallel = a)\}$, and (c) $\varphi_{i,2} = (X_{j_1}, X_{j_2}, X_{j_3} \rightarrow C_i, T_{pi,2})$ with $T_{pi,2} = \{(< a, \geq a, < a \parallel \neq a)\}$. Similarly, we can define the rest 5 CFD^{p}s $\varphi_{i,3}$, $\varphi_{i,4}$, $\varphi_{i,5}$, $\varphi_{i,6}$ and $\varphi_{i,7}$. Intuitively, we further use $< a$ and $\geq a$ to represent *false* and *true* for a variable, respectively.

(c) Σ_{n+1} contains a single CFD^p $\varphi_{n+1} = (Z \rightarrow Z, T_{p(n+1)})$ with $T_{p(n+1)} = \{(\_ \parallel = a)\}$. Intuitively, φ_{n+1} requires that for any R tuple t, $t[Z] = a$.

Observe that Σ contains $8 * m + n + 2$ CFD^{p}s in total. Thus the reduction is in PTIME.

We now show that ϕ is satisfiable iff Σ is satisfiable.

Assume first that Σ is satisfiable, then we show that there exists a nonempty instance I of R such that $I \models \Sigma$. For any tuple $t \in I$, (a) Σ_{n+1} forces $t[Z] = a$, (b) Σ_0 forces $t[C_1, \ldots, C_n] = (a, \ldots, a)$, and (c) for each clause C_i ($i \in [1, n]$) with variables $X_{j_1}, X_{j_2}, X_{j_3}$, Σ_j forces that $t[X_{j_1}, X_{j_2}, X_{j_3}]$ does not match the LHS of the CFD^{p}s that force $t[C_i] \neq a$. From the tuple t, we can construct a truth assignment ξ of ϕ such that $\xi(x_i)$ = *false* if $t[X_i] < a$ and $\xi(x_i)$ = *true* if $t[X_i] \geq a$ ($i \in [1, m]$). Since $\{t\} \models \Sigma$, it is easy to verify that the truth assignment ξ makes ϕ true.

Conversely, if ϕ is satisfiable, there exists a truth assignment ξ that makes ϕ true. We construct a tuple t of R as follows: (a) $t[C_1, \ldots, C_n, Z] = (a, \ldots, a)$ and (b) for each $i \in [1, m]$, $t[X_i] = a_i$ such that (a) $a_i \in$ dom and $a_i \geq a$ if $\xi(x_i)$ = *true* and (b) $a_i < a$ otherwise. Let $I = \{t\}$, then one can easily verify that $I \models \Sigma$.

Putting these together, we have the conclusion. □

Satisfiability analysis of CIND^{p}s. Like FDs, one can specify arbitrary INDs or CINDs without worrying about their satisfiability. Below we show that CIND^{p}s preserve this nice property, by extending the proof of its counterpart in [30].

Proposition 3: *Any set of* CIND^p*s is always satisfiable.* □

Proof: Given a set Σ of CIND^{p}s over a database schema $\mathcal{R}(R_1, \ldots, R_n)$, we show that one can always construct a *nonempty* instance D of $\mathcal{R}$ such that $D \models \Sigma$.

We build D as follows. First, for each attribute A, define the active domain of A to be a set $\mathsf{adom}(A)$, which consists of certain data values in $\mathsf{dom}(A)$. Second, using these active domains, we construct D.

(1) We start with the construction of active domains. (a) For each attribute A, initialize $\mathsf{adom}(A)$ along the same lines as the one for CFD^{p}s in Proposition 1; (b) For each CIND^p $(R_a[A_1, A_2, \ldots, A_m; X_p] \subseteq R_b[B_1, \ldots, B_m; Y_p], T_p)$ in Σ, let $\mathsf{adom}(B_i) = \mathsf{adom}(B_i) \cup \mathsf{adom}(A_i)$ for each $i \in [1, m]$, and this rule is repeatedly applied until a fixpoint of $\mathsf{adom}(A)$ is reached for all attributes A in $\mathcal{R}$.

It is easy to verify that this process always terminates as we start with a finite set of data values.

(2) We next construct the database instance D. For each relation $R_i(A_1, \ldots, A_k) \in \mathcal{R}$, we define $I_i = \mathsf{adom}(A_1) \times \cdots \times \mathsf{adom}(A_k)$, where $\times$ is the *Cartesian Product* operation [3]. Let $D = \{I_1, \ldots, I_n\}$, then it is easy to verify that D is nonempty and $D \models \Sigma$. □

Satisfiability analysis of CFD^{p}s and CIND^{p}s. The satisfiability problem for CFDs and CINDs taken together is undecidable [30]. Since CFD^{p}s and CIND^{p}s subsume CFDs and CINDs, respectively, we immediately have the following.

Corollary 4: *The satisfiability problem for* CFD^p*s and* CIND^p*s is undecidable.* □

4.2 Implication Analyses

The implication problem is to determine, given a set Σ of dependencies and another dependency ϕ, whether or not Σ entails ϕ, denoted by $\Sigma \models \phi$. That is, whether or not for all databases D, if $D \models \Sigma$ then $D \models \phi$.

The implication analysis helps us remove redundant rules, and thus improve the performance of error detection and repairing based on the rules [22], [30].

Example 7: The CFD^{p}s in Fig. 2 imply another CFD^p φ = item (sale, price $\rightarrow$ shipping, T), where T consists of a single pattern tuple (sale ='F', price $= 30 \parallel$ shipping $= 6$). Thus in the presence of the CFD^{p}s in Fig. 2, φ is redundant. □

Implication analysis of CFD^{p}s. We first show that the implication problem for CFD^{p}s retains the same complexity as their CFDs counterpart, verified by extending the proof of its counterpart in [22].

Proposition 5: *The implication problem for* CFD^p*s is co*NP*-complete.* □

Proof: The lower bound follows from the coNP-hardness of their CFDs counterpart [22], since CFDs are a special case of CFD^{p}s. The coNP upper bound is verified by presenting an NP algorithm for its complement problem for determining whether $\Sigma \not\models \varphi$.

We next present the a NP algorithm for its complement problem. The algorithm is based on a small model property: if $\varphi = R(X \rightarrow Y, T_p)$ and $\Sigma \not\models \varphi$, then there exists an instance I of R with two tuples t_1 and t_2 such that $I \models \Sigma$ and $t_1[X] = t_2[X] \asymp T_p[X]$, but either $t_1[Y] \neq t_2[Y]$ or $t_1[Y] \not\asymp T_p[Y]$ (resp. $t_2[Y] \not\asymp T_p[Y]$). Thus it suffices to consider instances I with two tuples only for deciding whether $\Sigma \not\models \varphi$.

Assume that the attributes $\mathsf{attr}(R) = \{A_1, \ldots, A_m\}$. For each $i \in [1, m]$, let $\mathsf{adom}(A_i)$ be the active domain defined in Proposition 1. Then one can easily verify that $\Sigma \not\models \varphi$ iff there exist two mappings ρ_1 and ρ_2 from all attributes A_i to $\mathsf{adom}(A_i)$ ($i \in [1, m]$) such that $I = \{(\rho_1(A_1), \ldots, \rho_1(A_m)), (\rho_2(A_1), \ldots, \rho_2(A_m))\}$, $I \models \Sigma$, but $I \not\models \varphi$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

Based on these, we give an NP algorithm as follows: (1) Guess two R tuples t_1 and t_2 such that $t_1[A_i], t_2[A_i] \in \mathsf{adom}(A_i)$ for each $i \in [1, m]$. (2) Check whether $I = \{t_1, t_2\}$ satisfies Σ, but not φ. If so the algorithm returns 'yes', and otherwise it repeats steps (1) and (2). Obviously step (2) can be done in PTIME in the size of Σ. Hence the algorithm is in NP, and so is the problem. □

Similar to the satisfiability analysis, it is known [22] that the implication analysis of CFDs is in PTIME when the CFDs are defined only with attributes that have an infinite domain. Analogous to Theorem 2, the result below shows that this is no longer the case for CFD^{p}s, which does not find a counterpart in [22].

Proposition 6: *In the absence of finite domain attributes, the implication problem for* CFD^p*s is co*NP*-complete.* □

Proof: The problem is in coNP by Proposition 5. The coNP-hardness is shown by reduction from the 3SAT problem to its complement problem, *i.e.,* the problem for determining whether $\Sigma \not\models \varphi$.

We next show the reduction from the 3SAT problem to the complement problem of the implication problem for CFD^{p}s, where 3SAT is NP-complete (cf. Proposition 2). Given an instance ϕ of 3SAT, we construct a relational schema R and a set $\Sigma \cup \{\varphi\}$ of CFD^{p}s defined on R such that ϕ is satisfiable iff $\Sigma \not\models \varphi$.

The relational schema R and the set Σ of CFD^{p}s are the same as the corresponding ones in Proposition 2. Moreover, φ is defined as $(Z \rightarrow Z, T_p)$, where $T_p = \{(\_ \; \| \neq a)\}$. Intuitively, φ requires that for any R tuple t, $t[Z] \neq a$. Along the same lines as Proposition 2, one can easily verify that ϕ is satisfiable iff $\Sigma \not\models \varphi$. Thus the problem is coNP-hard. □

Implication analysis of CIND^{p}s. We next show that CIND^{p}s do not make their implication analysis harder, verified by extending the proof of their CINDs counterpart given in [30].

Proposition 7: *The implication problem for* CIND^p*s is* EXPTIME-*complete.* □

Proof: The implication problem for CINDs is EXPTIME-hard [30]. Since CIND^{p}s subsume CINDs, the lower bound carries over to CIND^{p}s immediately. The EXPTIME upper bound is shown by presenting an EXPTIME algorithm that, given a set $\Sigma \cup \{\psi\}$ of CIND^{p}s over a database schema $\mathcal{R}$, determines whether $\Sigma \models \psi$ or not.

We next present the EXPTIME algorithm. Consider $\mathcal{R} = (R_1, \ldots, R_n)$ and $\psi = (R_a[X; X_p] \subseteq R_b[Y; Y_p], T_p)$. And for each attribute A, define the active domain $\mathsf{adom}(A)$ of A based on $\Sigma \cup \{\psi\}$ along the same line as the proof of Proposition 3. One can easily verify that if $\Sigma \not\models \psi$, there exists a non-empty instance D of $\mathcal{R}$ such that (a) $D \models \Sigma$ and $D \not\models \psi$, and (b) D consists of data values from the active domains only.

The detailed EXPTIME algorithm is given as follows.

(1) We first build a labeled directed graph $G(V, E, l)$. Each node $u \in V$ is a possible tuple '$R_i : t_i$' such that $t_i[A] \in \mathsf{adom}(A)$ for each attribute $A \in \mathsf{attr}(R_i)$. There is an edge e = ('$R_i : t_i$', '$R_j : t_j$') in E iff there exists a CIND^p $\phi = (R_i[U; U_p] \subseteq R_j[V; V_p], T_{p_\phi})$ in Σ such that $t_i[U_p] \asymp T_{p_\phi}[U_p]$, $t_j[V] = t_i[U]$ and $t_j[V_p] \asymp T_{p_\phi}[V_p]$, and e is labeled with the CIND^p ϕ, *i.e.,* $\phi \in l(e)$. Note that an edge may have multiple labels.

(2) Let S_a be the set of nodes '$R_a : t_a$' such that $t_a[X_p] \asymp T_p[X_p]$, and S_b be the set of nodes '$R_b : t_b$' such that $t_b[Y_p] \asymp T_p[Y_p]$, respectively.

(3) For each node u = '$R_a : t_a$' in S_a, let G_u be the induced subgraph of G that contains all the nodes reachable from u, and exactly the edges that appear in G over the same set of nodes. We also refer to u as the root of G_u.

(4) For an induced subgraph G_u of G with root u = '$R_a : t_a$', we derive another graph G'_u by recursively removing edges as follows. For any v in G_u, if v has a child v' from which no nodes in '$R_b : t_b$' in S_b with $t_b[X] = t_b[Y]$ are reachable, then for all children v'' of v, we remove from labels $l(v, v'')$ all the labels in $l(v, v')$, and edge (v, v'') is removed when $l(v, v'')$ becomes empty.

(5) If there exists a subgraph G'_u derived from an induced subgraph G_u of G with root u = '$R_a : t_a$' such that no nodes '$R_b : t_b$' in S_b with $t_a[X] = t_b[Y]$ are reachable from u, we return 'no', and return 'yes', otherwise.

It can be verified that (a) if the algorithm returns 'no', we can construct an instance D such that $D \models \Sigma$, but not ψ, by collecting those tuples attached on the end nodes of edges whole labels become empty at setp 4; and (b) if the algorithm returns 'yes', there exist no instances D such that $D \models \Sigma$, but not ψ.

We next show that the above algorithm indeed runs in exponential time: (a) The number of nodes in graph G is bounded by the maximum number of tuples in a database instance on $\mathcal{R}$. Let $|\Sigma \cup \{\psi\}|$ be the size of Σ and ψ, and $|\mathcal{R}|$ be the sum of arities of all relations in $\mathcal{R}$. Then the number of tuples in a database instance is bounded by $O(|\Sigma \cup \{\psi\}|^{|\mathcal{R}|})$; (b) The number of nodes in sets S_a or S_b is bounded by the maximum number of tuples in a database too; (c) The induced subgraph and the reachability testing can be done in linear-time in the size of the input [18].

Putting all these together, we have shown that the algorithm runs in exponential time. And, hence, the problem is in EXPTIME. □

It is known [30] that the implication problem is PSPACE-complete for CINDs defined with infinite domain attributes. Similar to Theorem 6, below we show that this no longer holds for CIND^{p}s.

Theorem 8: *In the absence of finite domain attributes, the implication problem for* CIND^p*s remains* EXPTIME*-complete.* □

Proof: The problem is in EXPTIME by Proposition 7. The EXPTIME-hardness is shown by reduction from the implication problem for CINDs in the general setting, in which finite-domain attributes may be present, that is known to be EXPTIME-complete [30].

We next present the reduction from the implication problem for CINDs in the general setting. Given a set $\Sigma \cup \{\psi\}$ of CINDs defined on a database schema $\mathcal{R}$ $(R_1, \ldots, R_n)$, we construct another database schema $\mathcal{R}'(R'_1, \ldots, R'_n)$, in which each relation R'_i $(i \in [1, n])$ consists of infinite domain attributes only, and a set $\Sigma' \cup \{\psi'\}$ of CIND^{p}s on $\mathcal{R}'$ such that $\Sigma \models \psi$ iff $\Sigma' \models \psi'$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

Table 1
Summary of Complexity Results

Σ	General setting		Infinite domain only	
	Satisfiability	Implication	Satisfiability	Implication
CFDs [22]	NP-complete	coNP-complete	PTIME	PTIME
CFD^{p}s	NP-complete	coNP-complete	NP-complete	coNP-complete
CINDs [30]	$O(1)$	EXPTIME-complete	$O(1)$	PSPACE-complete
CIND^{p}s	$O(1)$	EXPTIME-complete	$O(1)$	EXPTIME-complete
CFDs + CINDs [30]	undecidable	undecidable	undecidable	undecidable
CFD^{p}s + CIND^{p}s	undecidable	undecidable	undecidable	undecidable

(1) We start with constructing $\mathcal{R}'$. For each $R_i(A_1, \ldots, A_k)$ of $\mathcal{R}$, we define $R'_i(A'_1, \ldots, A'_k)$ such that for each attribute $A'_j (j \in [1, k])$, let $\mathsf{dom}(A'_j) = \mathsf{dom}(A_j)$ if $\mathsf{dom}(A_j)$ is infinite, and let $\mathsf{dom}(A'_j)$ be integer, a totally ordered infinite domain, if $\mathsf{dom}(A_j)$ is finite. Moreover, we define a mapping $\rho_{i,j}$ for each finite domain $\mathsf{dom}(A_j) = \{a_1, \ldots, a_h\}$ to integer: (a) Randomly choose h consecutive integers $\{b_1, \ldots, b_h\}$ such that for each $i \in [1, h-1]$, $b_{i+1} = b_i + 1$. (b) We now define the mapping $\rho_{i,j}(a_i) = b_i$ for $i \in [1, h]$. Moreover, we require two extra integers $b_0 = b_1 - 1$ and $b_{h+1} = b_h + 1$, denoted as $\rho_{i,j}.b_0$ and $\rho_{i,j}.b_{h+1}$. Note that this is always doable. For clarity, we also denote $\rho_{i,j}$ as ρ when it is clear from the context.

(2) We next define Σ' and ψ' on $\mathcal{R}'$ based on the mappings defined above. For each CIND $(R_a[X; A_1, \ldots, A_{m1}] \subseteq R_b[Y; B_1, \ldots, B_{m2}], T_p)$ in Σ and each $t_p \in T_p$, we define another CIND^p $(R'_a[X'; A'_1, \ldots, A'_{m1}, X'_p] \subseteq R'_b[Y'; B'_1, \ldots, B'_{m2}, Y'_p], T'_p)$, where (a) X' (resp. $Y', A'_1, \ldots, A'_{m1}$ and $B'_1, \ldots, B'_{m2}$) corresponds to X (resp. $Y, A_1, \ldots, A_{m1}$ and $B_1, \ldots, B_{m2}$); (b) X'_p (resp. Y'_p) corresponds to those finite domain attributes in R_a (resp. R_b), but not in $A_1, \ldots, A_{m1}$ (resp. $B_1, \ldots, B_{m2}$); and (c) $T'_p = \{t'_{p1}, t'_{p2}, t'_{p3}\}$ such that for each attribute A' in $A'_1, \ldots, A'_{m1}$ or $B'_1, \ldots, B'_{m2}$, (i) $t'_{p1}[A'] = t_p[A]$ and $t'_{p2}[A'] = t'_{p3}[A']$ = '_' if $\mathsf{dom}(A)$ is infinite, and (ii) $t'_{p1}[A'] = \rho(t_p[A])$ and $t'_{p2}[A'] = t'_{p3}[A']$ = '_' if $\mathsf{dom}(A)$ is finite; and (iii) for the rest attributes A' in X'_p or Y'_p, $t'_{p1}[A']$ = '_', $t'_{p2}[A']$ = '$> \rho.b_0$', and $t'_{p3}[A']$ = '$< \rho.b_{h+1}$'. Similarly, we can define a set Σ_ψ of CIND^{p}s from ψ based on the mappings.

Finally, one can easily verify that $\Sigma \models \psi$ iff $\Sigma' \models \Sigma_\psi$, *i.e.*, $\Sigma' \models \psi'$ for each CIND^p $\psi' \in \Sigma_\psi$. Following from this, the problem is EXPTIME-hard. □

Implication analysis of CFD^{p}s and CIND^{p}s. When CFD^{p}s and CIND^{p}s are taken together, their implication analysis is beyond reach in practice. This is not surprising since the implication problem for FDs and INDs is already undecidable [3]. Since CFD^{p}s and CIND^{p}s subsume FDs and INDs, respectively, from the undecidability result for FDs and INDs, the corollary below follows immediately.

Corollary 9: *The implication problem for* CFD^p*s and* CIND^p*s is undecidable.* □

Remarks. Inference systems play an important role for the implication analyses [3]. For the inference system of CFD^{p}s and CIND^{p}s alone, we can readily extend the one for CFDs [22] and CINDs [30], respectively, by deliberately handling the entailment of ordered pattern values involved with built-in predicates and their interaction with the wildcard '_'. The details are left to interested readers. Note that it is easy to know that the implication analysis of CFD^{p}s and CIND^{p}s together is not finitely axiomatizable by Corollary 9.

Summary. The complexity bounds for reasoning about CFD^{p}s and CIND^{p}s are summarized in Table 1. To give a complete picture we also include in Table 1 the complexity bounds for the static analyses of CFDs and CINDs, taken from [22], [30]. The results tell us the following.

(1) Despite the increased expressive power, CFD^{p}s and CIND^{p}s do not complicate the static analyses in the general case: the satisfiability and implication problems for CFD^{p}s and CIND^{p}s have the same complexity bounds as their counterparts for CFDs and CINDs, taken separately or together.

(2) In the special case when CFD^{p}s and CIND^{p}s are defined with infinite domain attributes only, however, their static analyses do not get simpler, as opposed to their counterparts for CFDs and CINDs. That is, the increased expressive power of CFD^{p}s and CIND^{p}s comes at a price in this special case.

5 VALIDATION OF CFD^{p}S AND CIND^{p}S

If CFD^{p}s and CIND^{p}s are to be used as data quality rules, the first question we have to settle is how to effectively detect errors and inconsistencies as violations of these dependencies, by leveraging functionality supported by commercial DBMS. More specifically, consider a database schema $\mathcal{R} = (R_1, \ldots, R_n)$, where R_i is a relational schema for $i \in [1, n]$. The error detection problem is stated as follows.

The *error detection problem* is to find, given a set Σ of CFD^{p}s and CIND^{p}s defined on $\mathcal{R}$, and a database instance $D = (I_1, \ldots, I_n)$ of $\mathcal{R}$ as input, the subset $(I'_1, \ldots, I'_n)$ of D such that for each $i \in [1, n]$, $I'_i \subseteq I_i$ and each tuple in I'_i violates at least one CFD^p or CIND^p in Σ. We denote the set as $\mathsf{vio}(D, \Sigma)$, referred to it as *the violation set* of D *w.r.t.* Σ.

In this section we develop SQL-based techniques for error detection based on CFD^{p}s and CIND^{p}s. The main result of the section is as follows.

Theorem 10: *Given a set Σ of* CFD^p*s and* CIND^p*s defined on $\mathcal{R} = (R_1, \ldots, R_n)$ and a database instance D of $\mathcal{R}$, a set of* SQL *queries can be automatically generated such that (a) the collection of the answers to the* SQL *queries in D is $\mathsf{vio}(D, \Sigma)$, and (b) the number and size of the set of* SQL *queries depend only on the number n of relations and their arities in $\mathcal{R}$, regardless of Σ.* □

Let $\Sigma^i_{\mathsf{cfd^p}}$ be the set of all CFD^{p}s in Σ defined on the same relational schema R_i, and $\Sigma^{(i,j)}_{\mathsf{cind^p}}$ the set of all CIND^{p}s in Σ from R_i to R_j, for $i, j \in [1, n]$. We show the following. (a) The violation set $\mathsf{vio}(D, \Sigma^i_{\mathsf{cfd^p}})$ can be computed by *two* SQL queries. (b) Similarly, $\mathsf{vio}(D, \Sigma^{(i,j)}_{\mathsf{cind^p}})$ can be computed by a *single* SQL query. (c) These SQL queries use pattern tableaux of CFD^{p}s (CIND^{p}s) encoded with data tables, and hence their sizes are independent of Σ. From these Theorem 10 follows immediately.

We next present the main techniques for the query generation method, and the key idea is to encode CFD^{p}s and CIND^{p}s with data tables so that data dependencies and data themselves are uniformly represented, and SQL queries are then automatically generated to detect those tuples that violate certain CFD^{p}s or CIND^{p}s.

5.1 Encoding CFD^{p}s and CIND^{p}s with Data Tables

We first show the following, by extending the encoding of [10], [22]. The pattern tableaux of all CFD^{p}s in $\Sigma^i_{\mathsf{cfd^p}}$ can be encoded with *three data tables*, and the pattern tableaux of all CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$ can be represented as *four data tables*, no matter how many dependencies are in the sets.

Encoding CFD^{p}s. We encode all pattern tableaux in $\Sigma^i_{\mathsf{cfd^p}}$ with three tables enc_L, enc_R and $\mathsf{enc}_{\neq}$, where enc_L (resp. enc_R) encodes the non-negation $(=, <, \leq, >, \geq)$ patterns in LHS (resp. RHS), and $\mathsf{enc}_{\neq}$ encodes those negation $(\neq)$ patterns. More specifically, we associate a unique id cid with each CFD^{p}s in $\Sigma^i_{\mathsf{cfd^p}}$, and let enc_L consist of the following attributes: (a) cid, (b) each attribute A appearing in the LHS of some CFD^{p}s in $\Sigma^i_{\mathsf{cfd^p}}$, and (c) its four companion attributes $A_>$, $A_\geq$, $A_<$, and $A_\leq$. That is, for each attribute, there are five columns in enc_L, one for each non-negation operator. Similarly, enc_R is defined. We use an $\mathsf{enc}_{\neq}$ tuple to encode a pattern $A \neq c$ in a CFD^p, consisting of cid, att, pos, and val, encoding the CFD^p id, the attribute A, the position ('LHS' or 'RHS'), and the constant c, respectively. Note that the arity of enc_L (enc_R) is bounded by $5 * |R_i| + 1$, where $|R_i|$ is the arity of R_i, and the arity of $\mathsf{enc}_{\neq}$ is 4.

Before we populate these tables, let us first describe a preferred form of CFD^{p}s that would simplify the analysis to be given. Consider a CFD^p $\varphi = R(X \rightarrow Y,\ T_p)$. If φ is not satisfiable we can simply drop it from Σ. Otherwise it is equivalent to a CFD^p $\varphi' = R(X \rightarrow Y,\ T'_p)$ such that for any pattern tuples t_p, t'_p in T'_p and for any attribute A in $X \cup Y$, (a) if $t_p[A]$ is op a and $t'_p[A]$ is op b, where op is not $\neq$, then $a = b$, and (b) if $t_p[A]$ is '_' then so is $t'_p[A]$. That is, for each non-negation op (resp. _), there is a *unique* constant a such that $t_p[A]$ = 'op a' (resp. $t_p[A] = \_$) is the only op (resp. _) pattern appearing in the A column of T'_p. We refer to $t_p[A]$ as $T'_p(\mathsf{op}, A)$ (resp. $T'_p(\_, A)$), and consider *w.l.o.g.* CFD^{p}s of this form only. Note that there are possibly multiple $t_p[A] \neq c$ patterns in T'_p,

We populate enc_L, enc_R and $\mathsf{enc}_{\neq}$ as follows. For each CFD^p $\varphi = R(X \rightarrow Y,\ T_p)$ in $\Sigma^i_{\mathsf{cfd^p}}$, we generate a distinct cid id_φ for it, and do the following.

(1) Add a tuple t_1 to enc_L such that (a) $t_1[\mathsf{cid}] = \mathsf{id}_\varphi$; (b) for each $A \in X$, $t_1[A]$ = '_' if $T'_p(\_, A)$ is '_', and for each non-negation predicate op, $t_1[A_{\mathsf{op}}]$ = 'a' if $T'_p(\mathsf{op}, A)$ is 'op a'; (c) we let $t_1[B]$ = **null** for all other attributes B in enc_L.
(2) Similarly add a tuple t_2 to enc_R for attributes in Y.
(3) For each attribute $A \in X \cup Y$ and each $\neq a$ pattern in $T_p[A]$, add a tuple t to $\mathsf{enc}_{\neq}$ such that $t[\mathsf{cid}] = \mathsf{id}_\varphi$, $t[\mathsf{att}]$ = 'A', $t[\mathsf{val}]$ = 'a', and $t[\mathsf{pos}]$ = 'LHS' (resp. $t[\mathsf{pos}]$ = 'RHS') if attribute A appears in X (resp. Y).

Example 8: Recall from Fig. 2 CFD^{p}s φ_2, φ_3 and φ_4 defined on relation item. The three CFD^{p}s are encoded with the tables shown in Fig. 4: (a) enc_L consists of attributes: cid, sale, price, price$_>$ and price$_\leq$; (b) enc_R consists of cid, shipping, price, price$_\geq$ and price$_<$; those attributes in a table with only 'null' pattern values do not contribute to error detection, and are thus omitted; And (c) $\mathsf{enc}_{\neq}$ is empty since all these CFD^{p}s have no negation patterns. One can easily reconstruct these CFD^{p}s from tables enc_L, enc_R and $\mathsf{enc}_{\neq}$ by collating the tuples based on cid. □

Encoding CIND^{p}s. All CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$ can be encoded with four tables enc, enc_L, enc_R and $\mathsf{enc}_{\neq}$. Here enc_L (resp. enc_R) and $\mathsf{enc}_{\neq}$ encode non-negation patterns on relation R_i (resp. R_j) and negation patterns on relations R_i or R_j, respectively, along the same lines as their counterparts for CFD^{p}s. We use enc to encode the INDs *embedded* in CIND^{p}s, which consists of the following attributes: (1) cid representing the id of a CIND^p, and (2) those X attributes of R_i and Y attributes of R_j appearing in some CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$. Note that the number of attributes in enc is bounded by $|R_i| + |R_j| + 1$, where $|R_i|$ is the arity of R_i.

For each CIND^p $\psi = (R_i[A_1 \ldots A_m;\ X_p] \subseteq R_j[B_1 \ldots B_m;\ Y_p],\ T_p)$ in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$, we generate a distinct cid id_ψ for it, and do the following.

(1) Add tuples t_1 and t_2 to enc_L and enc_R based on attributes X_p and Y_p, respectively, along the same lines as their CFD^p counterpart.
(2) Add tuples to $\mathsf{enc}_{\neq}$ in the same way as their CFD^p counterparts.
(3) Add tuple t to enc such that $t[\mathsf{cid}] = \mathsf{id}_\psi$. For each $k \in [1, m]$, let $t[A_k] = t[B_k] = k$, and $t[A]$ = **null** for the rest attributes A of enc.

Example 9: Figure 5 shows the coding of CIND^{p}s ψ_1 and ψ_2 given in Fig. 3. We use $\mathsf{state_L}$ and $\mathsf{state_R}$ in enc to denote the occurrences of attribute state in item and tax, respectively. In enc_L and enc_R, the attributes with only 'null' patterns are omitted, for the same reason as CFD^{p}s mentioned above. □

Putting these together, it is easy to verify that at most $O(n^2)$ data tables are needed to encode dependencies in Σ, regardless of the size of Σ. Recall that n is the number of relations in the database $\mathcal{R}$.

5.2 SQL-based Detection Methods

We next show how to generate SQL queries based on the encoding above. For each $i \in [1, n]$, we generate *two* SQL queries that, when evaluated on the I_i table of D, find $\mathsf{vio}(D, \Sigma^i_{\mathsf{cfd^p}})$. Similarly, for each $i, j \in [1, n]$, we generate a *single* SQL query $Q_{(i,j)}$ that, when evaluated on (I_i, I_j) of D, returns $\mathsf{vio}(D, \Sigma^{(i,j)}_{\mathsf{cind^p}})$. Putting these query answers together, we get $\mathsf{vio}(D, \Sigma)$, the violation set of D *w.r.t.* Σ.

SQL queries for CIND^{p}s. Below we show how the SQL query $Q_{(i,j)}$ is generated for validating CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$), which has not been studied by previous work. For the lack of space, we put the generation of detection queries for CFD^{p}s in the supplementary material, which is an extension of the SQL techniques for CFDs and eCFDs discussed in [22] and [10], respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

(1) enc_L

cid	sale	price	$\mathsf{price}_>$	$\mathsf{price}_\leq$
2	T	null	null	null
3	F	_	20	40
4	T	null	null	null

(2) enc_R

cid	shipping	price	$\mathsf{price}_\geq$	$\mathsf{price}_<$
2	0	null	null	null
3	6	null	null	null
4	null	_	2.99	9.99

(3) $\mathsf{enc}_{\neq}$

cid	pos	att	val

Figure 4. Encoding example of CFD^{p}s

(1) enc

cid	$\mathsf{state_L}$	$\mathsf{state_R}$
1	1	1
2	1	1

(2) enc_L

cid	type	state
1	_	null
2	_	DL

(3) enc_R

cid	rate
1	null
2	0

(4) $\mathsf{enc}_{\neq}$

cid	pos	att	val
1	LHS	type	art
2	LHS	type	art

Figure 5. Encoding example of CIND^{p}s

The query $Q_{(i,j)}$ for the validation of $\Sigma^{(i,j)}_{\mathsf{cind^p}}$ is given as follows, which capitalizes on the data tables enc, enc_L, enc_R and $\mathsf{enc}_{\neq}$ that encode CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$.

select $R_i.*$ **from** R_i, enc_L L, $\mathsf{enc}_{\neq}$ N
where $R_i.X \asymp L$ **and** $R_i.X \asymp N$ **and not exists** (
  select $R_j.*$ **from** R_j, enc H, enc_R R, $\mathsf{enc}_{\neq}$ N
  where $R_i.X = R_j.Y$ **and** L.cid = R.cid **and**
    L.cid = H.cid **and** $R_j.Y \asymp R$ **and** $R_j.Y \asymp N$)

Here (1) $X = \{\mathsf{A}_1, \ldots, \mathsf{A}_{\mathsf{m}1}\}$ and $Y = \{\mathsf{B}_1, \ldots, \mathsf{B}_{\mathsf{m}2}\}$ are the sets of attributes of R_i and R_j appearing in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$, respectively; (2) $R_i.X \asymp L$ is the conjunction of

$L.A_k$ **is null or** $R_i.A_k = L.A_k$ **or** ($L.A_k$ = '_' **and**
  ($L.A_{k_>}$ **is null or** $R_i.A_k > L.A_{k_>}$) **and**
  $L.A_{k_\geq}$ **is null or** $R_i.A_k \geq L.A_{k_\geq}$) **and**
  ($L.A_{k_<}$ **is null or** $R_i.A_k < L.A_{k_<}$) **and**
  ($L.A_{k_\leq}$ **is null or** $R_i.A_k \leq L.A_{k_\leq}$))

for each $k \in [1, m_1]$; (3) $R_j.Y \asymp R$ is defined similarly for attributes in Y; (4) $R_i.X \asymp N$ is a shorthand for the conjunction below, for each $k \in [1, m_1]$:

not exists (**select** $*$ **from** N
  where L.cid = N.cid **and** N.pos = 'LHS' **and**
    N.att = 'A_k' **and** $R_i.A_k$ = N.val);

(5) $R_j.Y \asymp N$ is defined similarly, but with N.pos = 'RHS' ; (6) $R_i.X = R_j.Y$ represents the following: for each A_k ($k \in [1, m_1]$) and each B_l ($l \in [1, m_2]$), ($H.A_k$ **is null or** $H.B_l$ **is null or** $H.B_l \neq H.A_k$ **or** $R_i.A_k = R_j.B_l$).

Intuitively, (1) $R_i.X \asymp L$ **and** $R_i.X \asymp N$ ensure that the R_i tuples selected match the LHS patterns of some CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$; (2) $R_j.Y \asymp R$ **and** $R_j.Y \asymp N$ check the corresponding RHS patterns of these CIND^{p}s on R_j tuples; (3) $R_i.X = R_j.Y$ enforces the *embedded* INDs; (4) L.cid = R.cid **and** L.cid = H.cid assure that the LHS and RHS patterns in the same CIND^p are correctly collated; and (5) **not exists** in $Q_{(i,j)}$ ensures that the R_i tuples selected violate CIND^{p}s in $\Sigma^{(i,j)}_{\mathsf{cind^p}}$.

Example 10: Using the coding of Fig. 5, an SQL query Q for checking CIND^{p}s ψ_1 and ψ_2 of Fig. 3 is given as follows:

select $R_1.*$ **from** item R_1, enc_L L, $\mathsf{enc}_{\neq}$ N
where (L.type **is null or** R_1.type = L.type **or** L.type = '_') **and**
  not exist (**select** * **from** N
    where N.cid = L.cid **and** N.pos = 'LHS' **and**
      N.att = 'type' **and** R_1.type = N.val) **and**
  (L.state **is null or** R_1.state = L.state **or** L.state = '_') **and**
  not exist (**select** * **from** N
    where N.cid = L.cid **and** N.pos = 'LHS' **and**
      N.att = 'state' **and** R_1.state =N.val) **and**
  not exist (**select** $R_2.*$ **from** tax R_2, enc H, enc_R R
    where ($H.\mathsf{state_L}$ **is null or** $H.\mathsf{state_R}$ **is null or**
      $H.\mathsf{state_L}$! = $H.\mathsf{state_R}$ **or** R_2.state = R_1.state)
      and L.cid = H.cid **and** L.cid = R.cid **and**
      (R.rate **is null or** R_2.rate = R.rate **or**
      R.rate = '_') **and not exist** (**select** * **from** N
      where N.cid = R.cid **and** N.pos = 'RHS'
      and N.att = 'rate' **and** R_2.rate =N.val))

The SQL queries generated can be simplified as follows. As shown in Example 10, when checking patterns imposed by enc, enc_L or enc_R, the queries need not consider attributes A if $t[A]$ is **null** for each tuple t in the table. Similarly, if an attribute A does not appear in any tuple in $\mathsf{enc}_{\neq}$, the queries need not check A either. From this, it follows that we do not even need to generate those attributes with only **null** patterns for data tables enc, enc_L or enc_R when encoding CIND^{p}s or CFD^{p}s. □

6 EXPERIMENTAL STUDY

We next present an extensive experimental study of CFD^{p}s and CIND^{p}s. Using real-life data, we conducted two sets of experiments to evaluate the efficiency and effectiveness of CFD^{p}s and CIND^{p}s vs. their counterparts CFDs and CINDs, separately and taken together.

6.1 Experimental Settings

We first present our experimental settings.

Datasets. We used two real-life datasets that were stored in an SQL Server 2012 database.

(1) HOSP (Hospital Compare) is a database publicly available from U.S. Department of Health & Human Services [1]. We used two tables hcahps and hcahps-state, which record the hospital level and state level ratings of the Hospital Consumer Assessment of Healthcare Providers and Systems (HCAHPS), respectively. For table hcahps, it records (a) the hospital information: hid (hospital ID), hname (hospital name), addr (address), city, state, zip, county, phn (phone number), and (b) the measure information: mid (measure ID), mq (question), mad (answer description), map (answer percentage), mncs (number of completed surveys), msrrp (survey response rate percentage), mfn (footnote). And for table hcahps-state, it records state level measure information: state, mid, mq and map, among other things.

We designed 6 CFD^{p}s and 3 CIND^{p}s for HOSP, shown below in an informal way for easy of understanding.

φ_1: hcahps (zip = '_' **and** city = '_' $\rightarrow$ state = '_')
φ_2: hcahps (hid = '_' $\rightarrow$ hname = '_' **and** county = '_' **and** addr = '_' **and** phn = '_')
φ_3: hcahps (hid = '_' $\rightarrow$ msrrp = '_')

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

φ_4: hcahps (mid = '_' $\rightarrow$ mq = '_' **and** mad = '_')
φ_5: hcahps (hid = '_' **and** mncs = 'Not Available' $\rightarrow$ mfn ≥ 1 **and** mfn ≤ 14)
φ_6: hcahps (hid = '_' **and** mid = '_' **and** mncs $\neq$ 'Not Available' **and** mncs $\neq$ 'Fewer than 100' $\rightarrow$ map ≥ 0 **and** map ≤ 100)
ψ_1: hcahps (mid, mq; nil) $\subseteq$ hcahps-state (mid, mq; nil)
ψ_2: hcahps (mid, state; nil) $\subseteq$ hcahps-state (mid, state; nil)
ψ_3: hcahps (mid, state; mncs $\neq$ 'Not Available') $\subseteq$ hcahps-state (mid, state; map ≥ 0 **and** map ≤ 100)

For comparison, we also designed the CFDs and CINDs counterparts of the above CFD^{p}s and CIND^{p}s. Here φ_1-φ_4 and ψ_1-ψ_2 are indeed CFDs and CINDs, respectively, while φ_5, φ_6 and ψ_3 are not. We hence further designed φ_5', φ_6' and ψ_3' to approximate φ_5, φ_6 and ψ_3, respectively.

φ_5': hcahps (hid = '_' **and** mncs = 'Not Available' $\rightarrow$ mfn = '_')
φ_6': hcahps (hid = '_' **and** mid = '_' **and** mncs = '_' $\rightarrow$ map = '_')
ψ_3': hcahps (mid, state; mncs = '_') $\subseteq$ hcahps-state (mid, state; map = '_')

(2) DBLP is a repository of computer science publications from 1946 to 2014 [2]. We further transformed its XML format into two tables paper and proceeding that record the paper and proceeding information, respectively, such that paper (key, type, title, booktitle, year, crossref, isbn, publisher) records books, journal articles and conference papers, and proceeding (key, year, isbn, publisher) records the proceedings of conference papers, respectively.

We generated 3482 CFD^{p}s and 2568 CIND^{p}s for the DBLP data, with their representatives shown below.

ϕ_1: paper (isbn = '_' **and** booktitle = '_' $\rightarrow$ publisher = '_')
ϕ_2: paper (title = '_' **and** year = '_' **and** booktitle = '_' $\rightarrow$ type = '_')
ϕ_3: paper (booktitle = 'CleanDB' $\rightarrow$ year = 2006)
ϕ_4: paper (booktitle = 'VLDB' **and** year$_\mathsf{L}$ = '_' $\rightarrow$ year$_\mathsf{R}$ $\geq$ 1975 **and** year$_\mathsf{R}$ $\leq$ 2007)
ϕ_5: paper (booktitle = 'PVLDB' **and** year$_\mathsf{L}$ = '_' $\rightarrow$ year$_\mathsf{R}$ $\geq$ 2008)
ρ_1: paper (crossref, isbn, publisher; type = 'inproceedings' **and** booktitle = 'CIKM-CNIKM') $\subseteq$ proceeding (key, isbn, publisher; year = 2009)
ρ_2: paper (crossref, isbn, publisher; type = 'inproceedings' **and** booktitle = 'VLDB') $\subseteq$ proceeding (key, isbn, publisher; year $\geq$ 1975 **and** year $\leq$ 2007)
ρ_3: paper (crossref, isbn, publisher; type = 'inproceedings' **and** booktitle = 'ICDE') $\subseteq$ proceeding (key, isbn, publisher; year $\geq$ 1984)

We collected all the booktitle and corresponding year from DBLP to generate the other CFD^{p}s and CIND^{p}s by instantiating the values of their booktitle and year attributes. Observe that ϕ_1-ϕ_3 and ρ_1 are CFDs and CINDs, respectively. For comparison, we further designed the following CFDs and CINDs to approximate ϕ_4-ϕ_5 and ρ_2-ρ_3.

ϕ_4': paper (booktitle = 'VLDB' **and** year$_\mathsf{L}$ = '_' $\rightarrow$ year$_\mathsf{R}$ = '_')
ϕ_5': paper (booktitle = 'PVLDB' **and** year$_\mathsf{L}$ = '_' $\rightarrow$ year$_\mathsf{R}$ = '_')
ρ_2': paper (crossref, isbn, publisher; type ='inproceedings' **and** booktitle = 'VLDB') $\subseteq$ proceeding (key, isbn, publisher; year = '_')
ρ_3': paper (crossref, isbn, publisher; type = 'inproceedings' **and** booktitle = 'ICDE') $\subseteq$ proceeding (key, isbn, publisher; year = '_')

Implementation. All the experiments were run within an SQL Server 2012 database installed on a machine with an Intel Core i5 (3.1GHz) CPU and 8GB of RAM. Each test was repeated 5 times, and the average is reported here.

6.2 Experimental Results

We next present our findings. Three parameters were used in our tests: (1) $|I_1|$, the number of tuples in table hcahps of HOSP or paper of DBLP, (2) $|I_2|$, the number of tuples in table hcahps-state of HOSP or proceeding of DBLP, and (3) $noise\%$, the percentage of dirty tuples in table hcahps of HOSP or paper of DBLP, ranging from 0% to 9%. For easy of comparison, we deliberately dirty the tuples in hcahps of HOSP or paper of DBLP so that using the CFD^{p}s and CIND^{p}s together can detect all the dirty tuples. A clean copy of HOSP and DBLP is also kept to tell whether a tuple is dirty or clean.

6.2.1 Tests of Efficiency

In the first set of experiments, we evaluated the violation detection efficiency of CFD^{p}s and CIND^{p}s vs. their counterparts CFDs and CINDs, separately and taken together.

Exp-1.1: CFD^{p}s vs. CFDs. (1) To evaluate the impacts of $|I_1|$, we fixed $noise\% = 9\%$, and varied $|I_1|$ from 10K to 90K for HOSP (resp. from 100K to 900K for DBLP); And (2) to evaluate the impacts of $noise\%$, we fixed $|I_1| = 90K$ for HOSP (resp. $900K$ for DBLP), and varied $noise\%$ from 0% to 9%. The results are reported in Figures 6(a) and 6(c) and Figures 6(b) and 6(d), respectively.

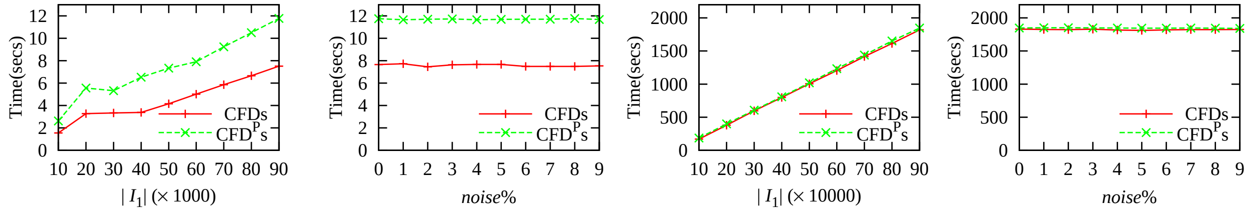
The results tell us that for CFDs and CFD^{p}s, both their running time (a) increases with the increment of the size of I_1, and (b) is insensitive to the noise. Furthermore, (c) their running time is mainly affected by three factors: the size of I_1, the LHS and RHS complexity of dependencies. For instance, (a) the LHS complexity of CFDs φ_5' and φ_6' is higher than CFD^{p}s φ_5 and φ_6, as they match more I_1 tuples, but the RHS complexity of CFDs φ_5' and φ_6' is lower than CFD^{p}s φ_5 and φ_6, as they are easier to check violations; And (b) the LHS complexity of CFDs ϕ_4' and ϕ_5' is the same as CFD^{p}s ϕ_4 and ϕ_5, but the RHS complexity of CFDs ϕ_4' and ϕ_5' is similar to CFD^{p}s ϕ_4 and ϕ_5, as they are easier to check violations. As a combined result, the running time of CFDs is lower than CFD^{p}s on HOSP, but close to CFD^{p}s on DBLP.

Exp-1.2: CIND^{p}s vs. CINDs. (1) To evaluate the impacts of $|I_1|$, we fixed $noise\%$ = 9% and $|I_2|$ = $1.6K$ for HOSP (resp. $16K$ for DBLP), and varied $|I_1|$ from $10K$ to $90K$ for HOSP (resp. from $100K$ to $900K$ for DBLP); (2) To evaluate the impacts of $|I_2|$, we fixed $noise\%$ = 9% and $|I_1|$ = $90K$ for HOSP (resp. $900K$ for DBLP), and varied $|I_2|$ from $1K$ to $1.6K$ for HOSP (resp. from $10K$ to $16K$ for DBLP); And (3) To evaluate the impacts of $noise\%$, we fixed $|I_1|$ = $90K$ for HOSP (resp. $900K$ for DBLP) and $|I_2|$ = $1.6K$ for HOSP (resp. $16K$ for DBLP), and varied $noise\%$ from 0% to 9%. The results are reported in Figures 7(a) and 7(d), Figures 7(b) and 7(e), and Figures 7(c) and 7(f), respectively.

The results tell us that for CINDs and CIND^{p}s, both their running time (a) increases with the increment of the size of I_1, (b) is not affected much by I_2 as $|I_2|$ is relatively small in the tests, and (c) is insensitive to the noise. Furthermore, (d) their running time is mainly affected by four factors: the size of I_1, the size of I_2, the LHS and RHS complexity of dependencies. For instance, (a) the LHS complexity of CIND ψ_3' is higher than CIND^p ψ_3, as they match more I_1 tuples, but the RHS complexity of CIND ψ_3' is lower than CIND^p ψ_3, as they are easier to check violations; And (b) the LHS complexity of CINDs ρ_2' and ρ_3' is the same as CIND^{p}s ρ_2 and ρ_3, but the RHS complexity of CINDs ρ_2' and ρ_3' is lower than CIND^{p}s ρ_2 and ρ_3, as they are easier to check violations. As a combined result, the running time of CINDs is close to CIND^{p}s on HOSP, but is lower on DBLP.

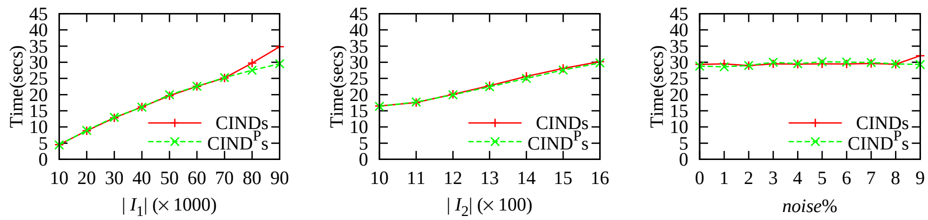
Exp-1.3: CFD^{p}s + CIND^{p}s vs. CFDs + CINDs. Using the same setting as Exp-1.2, we evaluated the impacts of $|I_1|$, $|I_2|$ and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

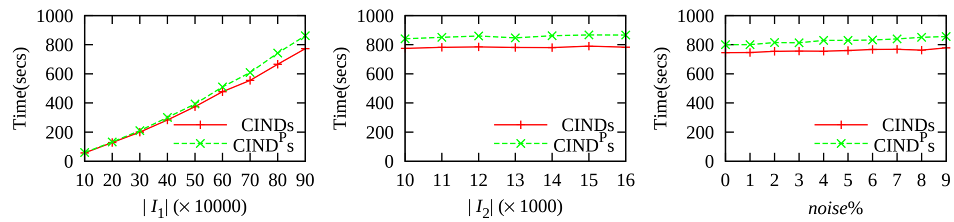


(a) Varying $|I_1|$ for HOSP (b) Varying $noise\%$ for HOSP (c) Varying $|I_1|$ for DBLP (d) Varying $noise\%$ for DBLP

Figure 6. Efficiency of detecting violations: CFD^{p}s vs. CFDs

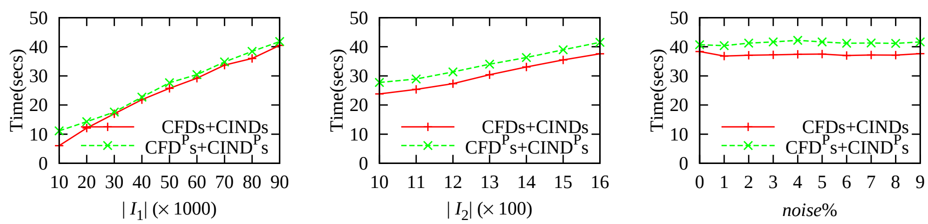


(a) Varying $|I_1|$ for HOSP (b) Varying $|I_2|$ for HOSP (c) Varying $noise\%$ for HOSP

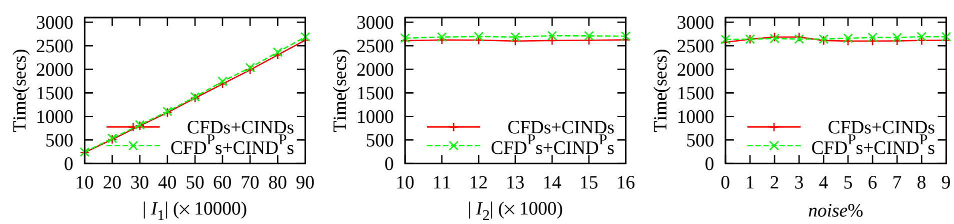


(d) Varying $|I_1|$ for DBLP (e) Varying $|I_2|$ for DBLP (f) Varying $noise\%$ for DBLP

Figure 7. Efficiency of detecting violations: CIND^{p}s vs. CINDs



(a) Varying $|I_1|$ for HOSP (b) Varying $|I_2|$ for HOSP (c) Varying $noise\%$ for HOSP



(d) Varying $|I_1|$ for DBLP (e) Varying $|I_2|$ for DBLP (f) Varying $noise\%$ for DBLP

Figure 8. Efficiency of detecting violations: CFD^{p}s + CIND^{p}s vs. CFDs + CINDs

$noise\%$. The results are reported in Figures 8(a) and 8(d), Figures 8(b) and 8(e) and Figures 8(c) and 8(f), respectively. The results show similar findings to Exp-1.1 and Exp-1.2, and are consistent with them.

6.2.2 Tests of Effectiveness

In the second set of experiments, we evaluated the violation detection effectiveness of CFD^{p}s and CIND^{p}s vs. their counterparts CFDs and CINDs, separately and taken together. Note that we did not report the results of varying $|I_2|$ as it has no impacts on the effectiveness tests in our setting.

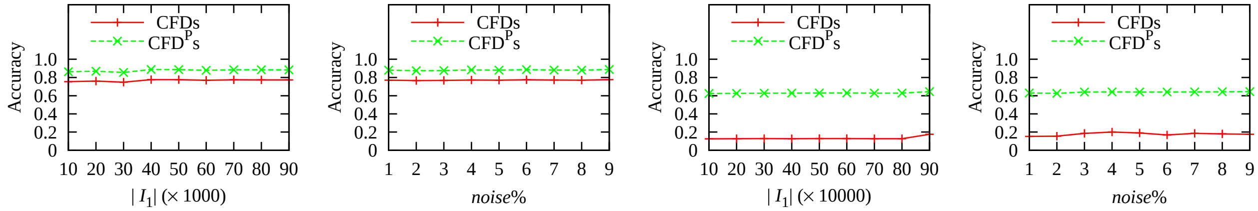
Given one of CFDs, CFD^{p}s, CINDs, CIND^{p}s, CFDs + CINDs or CFD^{p}s + CIND^{p}s, denoted by x, its effectiveness of detecting violations is evaluated with the following measure:

$$\text{accuracy}(x) = \frac{\#\text{dirty tuples found by } x}{\#\text{dirty tuples found by cfd}^p\text{s + cind}^p\text{s}}.$$

Exp-2. Using the same setting as Exp-1.1, Exp-1.2 and Exp-1.3, respectively, we evaluated the impacts of $|I_1|$ and $noise\%$ for (a) CFD^{p}s vs. CFDs, (b) CIND^{p}s vs. CINDs and (c) CFD^{p}s + CIND^{p}s vs. CFDs + CINDs, respectively. The results are reported in Figures 9, 10 and 11, respectively, and are summarized in Table 2.

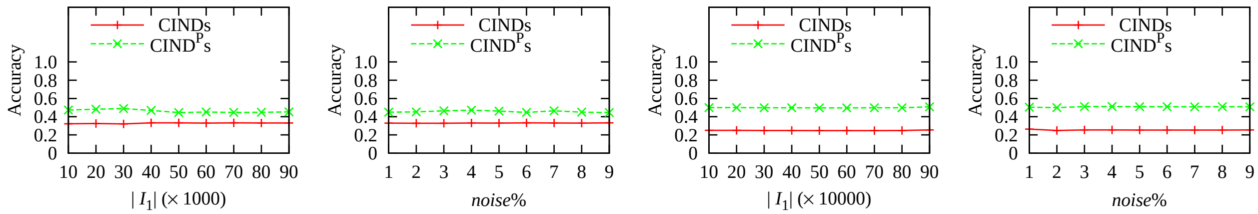
The results tell us that (1) the effectiveness of detecting violations using all classes of dependencies are robust to $|I_1|$ and $noise\%$, (2) CFD^{p}s, CIND^{p}s and CFD^{p}s + CIND^{p}s obviously outperform their counterparts CFDs, CINDs and CFDs + CINDs, respectively, (3) the increase of effectiveness

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering



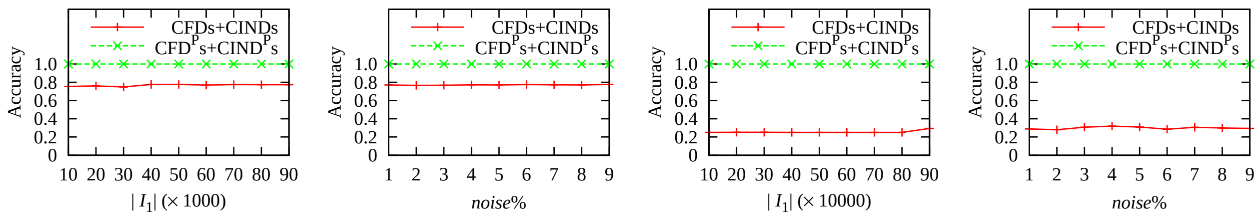
(a) Varying $|I_1|$ for HOSP (b) Varying $noise\%$ for HOSP (c) Varying $|I_1|$ for DBLP (d) Varying $noise\%$ for DBLP

Figure 9. Effectiveness of detecting violations: CFD^{p}s vs. CFDs



(a) Varying $|I_1|$ for HOSP (b) Varying $noise\%$ for HOSP (c) Varying $|I_1|$ for DBLP (d) Varying $noise\%$ for DBLP

Figure 10. Effectiveness of detecting violations: CIND^{p}s vs. CINDs



(a) Varying $|I_1|$ for HOSP (b) Varying $noise\%$ for HOSP (c) Varying $|I_1|$ for DBLP (d) Varying $noise\%$ for DBLP

Figure 11. Effectiveness of detecting violations: CFD^{p}s + CIND^{p}s vs. CFDs + CINDs

Table 2
Summary of violation detection accuracy

Datasets	varying	CFDs (%)	CFD^{p}s (%)	CINDs (%)	CIND^{p}s (%)	CFDs + CINDs (%)	CFD^{p}s + CIND^{p}s (%)
HOSP	$\|I_1\|$	[74.8, 77.7]	[85.5, 88.8]	[32.1, 33.3]	[44.6, 48.9]	[74.8, 77.7]	100
HOSP	noise%	[76.6, 77.7]	[87.5, 88.8]	[32.8, 33.3]	[44.5, 47.2]	[76.6, 77.7]	100
DBLP	$\|I_1\|$	[12.6, 17.6]	[62.5, 64.5]	[24.8, 25.4]	[49.6, 50.9]	[25.0, 29.5]	100
DBLP	noise%	[15.3, 20.1]	[62.5, 64.5]	[24.9, 26.3]	[50.0, 51.1]	[28.0, 31.9]	100

depends on the increase of the expressive power, and varies from 22% to 75% on HOSP and DBLP, and, (4) the increased effectiveness on DBLP is larger than on HOSP, as there are more CFD^{p}s and CIND^{p}s on HOSP that can be expressed by CFDs and CINDs than on DBLP in our tests.

Summary. From these experimental results on real-life data HOSP and DBLP, we find the following. (1) The running time of CFD^{p}s and CIND^{p}s is comparable to their CFDs and CINDs counterparts, which is consistent with the the static analyses: CFD^{p}s and CIND^{p}s retain the same complexity as their CFDs and CINDs counterparts. (2) CFD^{p}s and CIND^{p}s are able to capture more dirty tuples than CFDs and CINDs, due to the increased expressive power.

7 CONCLUSIONS

We have proposed CFD^{p}s and CIND^{p}s, which further extend CFDs and CINDs, respectively, by allowing patterns on data values to be expressed in terms of $\neq, <, \leq, >$ and $\geq$ predicates. We have shown that CFD^{p}s and CIND^{p}s are more powerful than CFDs and CINDs for detecting errors in real-life data. In addition, the satisfiability and implication problems for CFD^{p}s and CIND^{p}s have the same complexity bounds as their counterparts for CFDs and CINDs, respectively. We have also provided automated methods to generate SQL queries for detecting errors based on CFD^{p}s and CIND^{p}s. These provide commercial DBMS with an immediate capability to capture errors commonly found in real-world data.

One topic for future work is to develop a dependency language that is capable of expressing various extensions of CFDs (*e.g.*, CFD^{p}s, eCFDs [10] and CFD^{c}s [13]), without increasing the complexity of static analyses. Second, we are to develop effective algorithms for discovering CFD^{p}s and CIND^{p}s, along the same lines as [5], [28], [36]. Third, we plan to extend the methods of [8], [17] to repair data based on CFD^{p}s and CIND^{p}s, instead of using CFDs [17], traditional FDs and INDs [8], denial constraints [7], and aggregate constraints [25].

ACKNOWLEDGMENTS

Ma is supported in part by 973 program 2014CB340300 and NSFC 61322207. Fan is supported in part by NSFC

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2451632, IEEE Transactions on Knowledge and Data Engineering

61133002, 973 Program 2014CB340302, Shenzhen Peacock Program 1105100030834361, Guangdong Innovative Research Team Program 2011D005, EPSRC EP/J015377/1 and EP/M025268/1, and a Google Faculty Research Award.

REFERENCES

[1] *https://data.medicare.gov/data/hospital-compare.*

[2] *http://dblp.uni-trier.de/xml/.*

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

[4] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, 1999.

[5] J. Bauckmann, Z. Abedjan, U. Leser, H. Müller, and F. Naumann. Discovering conditional inclusion dependencies. In *CIKM*, 2012.

[6] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-Generating Dependencies. *J. Comput. Syst. Sci.*, 59(1):94–115, 1999.

[7] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008.

[8] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, 2005.

[9] P. D. Bra and J. Paredaens. Conditional dependencies for horizontal decompositions. In *ICALP*, 1983.

[10] L. Bravo, W. Fan, F. Geerts, and S. Ma. Increasing the expressivity of conditional functional dependencies without extra complexity. In *ICDE*, 2008.

[11] M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *J. Comput. Syst. Sci.*, 28(1):29–59, 1984.

[12] W. Chen, W. Fan, and S. Ma. Analyses and validation of conditional dependencies with built-in predicates. In *DEXA*, 2009.

[13] W. Chen, W. Fan, and S. Ma. Incorporating cardinality constraints and synonym rules into conditional functional dependencies. *IPL*, 109(14):783–789, 2009.

[14] F. Chiang and R. J. Miller. Discovering data quality rules. In *VLDB*, 2008.

[15] J. Chomicki. Consistent query answering: Five easy pieces. In *ICDT*, 2007.

[16] E. F. Codd. Relational completeness of data base sublanguages. In *Data Base Systems: Courant Computer Science Symposia Series 6*, pages 65–98. Prentice-Hall, 1972.

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

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[19] O. Curé. Improving the data quality of drug databases using conditional dependencies and ontologies. *J. Data and Information Quality*, 4(1):3, 2012.

[20] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma. Dynamic constraints for record matching. *VLDB J.*, 20(4):495–520, 2011.

[21] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[22] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *TODS*, 33(2), 2008.

[23] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *VLDB J.*, 21(2):213–238, 2012.

[24] W. Fan, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. *J. Data and Information Quality*, 4(4):16, 2014.

[25] S. Flesca, F. Furfaro, and F. Parisi. Consistent query answers on numerical databases under aggregate constraints. In *DBPL*, 2005.

[26] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[27] S. Ginsburg and R. Hull. Order dependency in the relational model. *Theor. Comput. Sci.*, 26:149–195, 1983.

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

[29] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric functional dependencies. In *ICDE*, 2009.

[30] S. Ma, W. Fan, and L. Bravo. Extending inclusion dependencies with conditions. *Theor. Comput. Sci.*, 515:64–95, 2014.

[31] M. J. Maher. Constrained dependencies. *TCS*, 173(1):113–149, 1997.

[32] M. J. Maher and D. Srivastava. Chasing Constrained Tuple-Generating Dependencies. In *PODS*, 1996.

[33] B. Saha and D. Srivastava. Data quality: The other face of big data. In *ICDE*, 2014.

[34] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *ACM Trans. Database Syst.*, 36(3):16, 2011.

[35] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1), 1997.

[36] A. Zanzi and A. Trombetta. Discovering non-constant conditional functional dependencies with built-in predicates. In *DEXA*, 2014.

PLACE PHOTO HERE

Shuai Ma is a professor at the School of Computer Science and Engineering, Beihang University. He obtained his PhD degrees from University of Edinburgh in 2010, and from Peking University in 2004, respectively. He was a postdoctoral research fellow in the database group, University of Edinburgh, a summer intern at Bell labs, Murray Hill, USA, in the summer of 2008, and a visiting researcher of MRSA in 2012. He is a recipient of the Best Paper Award for VLDB 2010 and the Best Challenge Paper Award for WISE 2013. His current research interests include database theory and systems, social data analysis and data intensive computing.

PLACE PHOTO HERE

Liang Duan is a PhD student in the School of Computer Science and Engineering, Beihang University, co-supervised by Prof. Jinpeng Huai and Prof. Shuai Ma. He received his MS degree in computer software and theory from Yunnan University in 2014, and BS degree in computer science and technology from Beihang University in 2009. His current research interests include databases and social data analysis.

PLACE PHOTO HERE

Wenfei Fan is Professor (Chair) of Web Data Management in the School of Informatics, University of Edinburgh, UK. He is a Fellow of ACM, a Fellow of the Royal Society of Edinburgh, UK, a National Professor of the Thousand-Talent Program and a Yangtze River Scholar, China. He received his PhD degree from the University of Pennsylvania. He is a recipient of the Runner-up for Best Paper Award of ICDE 2014, the Alberto O. Mendelzon Test-of-Time Award of ACM PODS 2010, the Best Paper Award for VLDB 2010, the Roger Needham Award in 2008 (UK), the Best Paper Award for ICDE 2007, the Best Paper of the Year Award for Computer Networks in 2002, and the Career Award in 2001 (USA). His current research interests include database theory and systems.

PLACE PHOTO HERE

Chunming Hu is an associate professor at the School of Computer Science and Engineering, Beihang University. He received his PhD degree from Beihang University in 2006. His current research interests include distributed systems, system virtualization, large scale data management and processing systems.

PLACE PHOTO HERE

Wenguang Chen is an associate professor at the Department of Computer Science, Peking University. He obtained his PhD degree from Peking University in 2009. He was a visiting scholar at University of Alberta from 2011 to 2012 and at University of Hawaii at Manoa in from 2012 to 2013. His current research interests include data management, data quality and intelligent HCI.