

# 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 [1]) and conditional inclusion dependencies (CINDs [2]), 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  $\Sigma$  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  $\Sigma$ . 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 [1]) and *conditional inclusion dependencies* (CINDs [2]), 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 [2], [3], [4].

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. [5]) and conditional dependencies (CFDs, CINDs [1], [2]) on the data include the following:

$cf_{d1}$ : *item* ( $id \rightarrow name, type, price, shipping, sale$ )  
 $cf_{d2}$ : *tax* ( $state \rightarrow rate$ )  
 $cf_{d3}$ : *item* ( $sale = 'T' \rightarrow shipping = 0$ )

These are CFDs: (a)  $cf_{d1}$  assures that the id of an item uniquely determines its name, type, price, shipping and sale; (b)  $cf_{d2}$  states that *state* is a key for *tax*, i.e., for each state there is a unique sale tax rate; and (c)  $cf_{d3}$  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  $cf_{d3}$  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,  $cf_{d1}$  and  $cf_{d2}$  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  $cf_{d1}$ ,  $cf_{d2}$  and  $cf_{d3}$ . 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

- S. Ma, L. Duan, and C. Hu are with the SKLSE 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, United Kingdom. E-mail: wenfei@inf.ed.ac.uk.
- W. Chen is with the Department of Information Management, Peking University, China. E-mail: chenwog@pku.edu.cn.

Manuscript received 8 Dec. 2014; revised 4 May 2015; accepted 19 June 2015.  
 Date of publication 30 June 2015; date of current version 3 Nov. 2015.

Recommended for acceptance by X. Lin.

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.

Digital Object Identifier no. 10.1109/TKDE.2015.2451632

|         | 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

Fig. 1. Example instance  $D_0$  of item and tax.

relations cannot be expressed as CFDs of [1] or CINDs of [2], but can be expressed as the following dependencies:

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

Here  $\text{pfd}_2$  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  $\text{pfd}_1$  and  $\text{pfd}_3$ . These dependencies extend CFDs [1] by specifying patterns of semantically related data values in terms of predicates  $<$ ,  $\leq$ ,  $>$  and  $\geq$ . Similarly,  $\text{pfd}_4$  assures that for any item tuple, if it is on sale, then its price must be in the range [2.99, 9.99). Finally,  $\text{pind}_1$  extends CINDs [2] by specifying patterns with  $\neq$ : for any item tuple  $t$ , if  $t[\text{type}]$  is not artwork, then there must exist a tax tuple  $t'$  such that  $t[\text{state}] = t'[\text{state}]$ , i.e., the sale tax of the item can be found from the tax relation.

Using dependencies  $\text{pfd}_1$ – $\text{pfd}_4$  and  $\text{pind}_1$  as data quality rules, we find that  $D_0$  of Fig. 1 is not clean. Indeed, (a)  $t_2$  violates  $\text{pfd}_1$ : its price is less than 20, but its shipping fee is 2 rather than 3; similarly,  $t_3$  violates  $\text{pfd}_2$ , and  $t_4$  violates  $\text{pfd}_3$ . (b) Tuple  $t_1$  violates  $\text{pfd}_4$ : it is on sale but its price is not in the range [2.99, 9.99). (c) The database  $D_0$  also violates  $\text{pind}_1$ :  $t_1$  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  $\text{pfd}_1$ – $\text{pfd}_4$  and  $\text{pind}_1$  can be expressed as FDs or INDs [5], which do not allow constants, or as CFDs [1] or CINDs [2], which specify patterns with equality (=) only. While there have been extensions of CFDs [6], [7], [8], 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 [9] 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.

## 1.1 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<sup>p</sup>s and CIND<sup>p</sup>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<sup>p</sup>s or CIND<sup>p</sup>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<sup>p</sup>s and CIND<sup>p</sup>s, taken separately or together (Section 4). The satisfiability problem is to determine whether a set  $\Sigma$  of dependencies has a nonempty model, i.e., whether the rules in  $\Sigma$  are consistent themselves. The implication problem is to decide whether a set  $\Sigma$  of dependencies entails another dependency  $\phi$ , i.e., whether the rule  $\phi$  is redundant in the presence of the rules in  $\Sigma$ . These are the central technical problems associated with any dependency language.

We show that despite the increased expressive power, CFD<sup>p</sup>s and CIND<sup>p</sup>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<sup>p</sup>s, respectively, (b) in  $O(1)$ -time (constant-time) and EXPTIME-complete for CIND<sup>p</sup>s, respectively, and (c) are undecidable when CFD<sup>p</sup>s and CIND<sup>p</sup>s are taken together. These are the same as their CFDs and CINDs counterparts [2]. In contrast, data with linearly ordered domains often makes our lives harder [10].

- 3) We provide SQL-based techniques to detect errors based on CFD<sup>p</sup>s and CIND<sup>p</sup>s (Section 5). Given a set  $\Sigma$  of CFD<sup>p</sup>s and CIND<sup>p</sup>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  $\Sigma$ . Further, the SQL queries are independent of the size and cardinality of  $\Sigma$ . These provide the capability of detecting errors in a single relation (CFD<sup>p</sup>s) and across different relations (CIND<sup>p</sup>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<sup>p</sup>s and CIND<sup>p</sup>s is comparable to their CFDs and CINDs counterparts, which is consistent with the static analyses in Section 4, and (b) CFD<sup>p</sup>s and CIND<sup>p</sup>s are able to capture more errors than their CFDs and CINDs counterparts (22 percent on HOSP and 75 percent on DBLP), due to the increased expressive power.

## 1.2 Related Work

This paper is an extension of our earlier work [9] by adding (a) the proofs for the complexity bounds for the

satisfiability and implication analyses of CFD<sup>p</sup>s and CIND<sup>p</sup>s, separately and taken together (Section 4), and (b) an extensive experimental study of CFD<sup>p</sup>s and CIND<sup>p</sup>s (Section 6), which was not investigated in [9].

Recently, data dependencies have generated renewed interests for improving data quality [1], [2], [6], [7], [11], [12], [13], [14], [15]. Constraint-based data cleaning was introduced in [16], which proposed to use dependencies, e.g. FDs, INDS and denial constraints, to detect and repair errors in real-life data (see, e.g. [5], [11], [15] for details). Data dependencies have been studied for relational databases since the introduction of FDs by Codd [17] in 1972 (see, e.g. [5] for details), and the theory of INDS was established in [18], 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 [1], for improving the quality of data. It was shown in [1] that the satisfiability and implication problems for CFDs are NP-complete and coNP-complete, respectively. Along the same lines, CINDs [2] were proposed to extend INDS, and it was shown [2] that the satisfiability and implication problems for CINDs are in constant time and EXPTIME-complete, respectively. SQL techniques were developed in [1] to detect errors by using CFDs, but have not been studied for CINDs. This work extends the static analyses of conditional dependencies of [1], [2], 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<sup>p</sup>s and CIND<sup>p</sup>s taken together.

Extensions of CFDs have been proposed to support disjunction and negation [6], cardinality constraints and synonym rules [8], and to specify patterns in terms of value ranges [7]. While CFD<sup>p</sup>s are more powerful than the extension of [7], they cannot express disjunctions [6], cardinality constraints and synonym rules [8]. 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 [7], [12], CFD<sup>p</sup>s [13] and CINDs [14] and for repairing data based on either CFDs [3], traditional FDs and INDS taken together [19], CFDs and CINDs taken together [20], denial constraints [21], aggregate constraints [22], matching dependencies [23], matching dependencies and CFDs [24], or editing rules and master data [25]. We defer the treatment of these topics for CFD<sup>p</sup>s and CIND<sup>p</sup>s to future work.

A variety of extensions of FDs and INDS have been studied for specifying constraint databases and constraint logic programs [26], [27], [28], [29], [30]. While the languages of [26], [27], [29] cannot express CFD<sup>p</sup>s, constraint-generating dependencies (CGDs) of [27] and constrained tuple-generating dependencies (CTGDs) of [30] can express CFD<sup>p</sup>s, and CTGDs can also express CIND<sup>p</sup>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 [26] supporting the comparison of attributes with  $=$ ,  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ , matching dependencies [23] and differential dependencies [31] that support the comparison of attributes with  $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$ ,  $\geq$  for record matching. However, different from CFDs and CFD<sup>p</sup>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 [32].

## 2 EXTENDING CFDs WITH PREDICATES

We now define *conditional functional dependencies with predicates*, denoted by CFD<sup>p</sup>s, by extending CFDs [1] 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  $\text{attr}(R)$ . For each attribute  $A \in \text{attr}(R)$ , its domain is specified in  $R$ , denoted as  $\text{dom}(A)$ , 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.

### 2.1 Syntax

A CFD<sup>p</sup>  $\varphi$  on  $R$  is a pair  $R(X \rightarrow Y, T_p)$ , where (1)  $X, Y$  are sets of attributes in  $\text{attr}(R)$ ; (2)  $X \rightarrow Y$  is a standard FD, referred to as the FD *embedded in*  $\varphi$ ; and (3)  $T_p$  is a tableau with attributes in  $X$  and  $Y$ , referred to as the *pattern tableau* of  $\varphi$ , 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  $\text{dom}(A)$ , or ' $\text{op } a$ ', where  $\text{op}$  is one of  $\{=, \neq, <, \leq, >, \geq\}$ , and ' $a$ ' is a constant in  $\text{dom}(A)$ .

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 ' $\parallel$ '. We simply write  $\varphi$  as  $(X \rightarrow Y, T_p)$  when  $R$  is clear from the context, and denote  $X$  as  $\text{LHS}(\varphi)$  and  $Y$  as  $\text{RHS}(\varphi)$ , respectively.

**Example 2.** The dependencies  $\text{cfd}_1$ – $\text{cfd}_3$  and  $\text{pfd}_1$ – $\text{pfd}_4$  that we have seen in Example 1 can all be expressed as CFD<sup>p</sup>s. Some of these CFD<sup>p</sup>s are illustrated in Fig. 2, in which  $\varphi_1$  is for FD  $\text{cfd}_2$ ,  $\varphi_2$  is for CFD  $\text{cfd}_3$ ,  $\varphi_3$  is for  $\text{pfd}_2$ , and  $\varphi_4$  is for  $\text{pfd}_4$ , respectively.

### 2.2 Semantics

Consider CFD<sup>p</sup>  $\varphi = R(X \rightarrow Y, T_p)$ , where  $T_p = \{t_{p_1}, \dots, t_{p_k}\}$ .

A data tuple  $t$  of  $R$  is said to *match*  $\text{LHS}(\varphi)$ , denoted by  $t[X] \preceq_{T_p} [X]$ , if for each tuple  $t_{p_i}$  ( $i \in [1, k]$ ) in  $T_p$  and each attribute  $A$  in  $X$ , either (a)  $t_{p_i}[A]$  is the wildcard ' $'_ '$ ' (which matches any value in  $\text{dom}(A)$ ), or (b)  $t[A] \text{ op } a$  if  $t_{p_i}[A]$  is ' $\text{op } a$ ', where the operator  $\text{op}$  ( $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$  or  $\geq$ ) is interpreted by its standard semantics. Similarly, the notion that  $t$  *matches*  $\text{RHS}(\varphi)$  is defined, denoted by  $t[Y] \preceq_{T_p} [Y]$ .

Intuitively, each pattern tuple  $t_{p_i}$  ( $i \in [1, k]$ ) specifies a condition via  $t_{p_i}[X]$ , and  $t[X] \preceq_{T_p} [X]$  if  $t[X]$  satisfies the



(1)  $\varphi_1 = \text{tax}(\text{state} \rightarrow \text{rate}, T_1)$

|       |      |
|-------|------|
| state | rate |
| —     | —    |

$T_1$ :

(2)  $\varphi_2 = \text{item}(\text{sale} \rightarrow \text{shipping}, T_2)$

|      |          |
|------|----------|
| sale | shipping |
| = T  | = 0      |

$T_2$ :

(3)  $\varphi_3 = \text{item}(\text{sale}, \text{price} \rightarrow \text{shipping}, T_3)$

|      |       |          |
|------|-------|----------|
| sale | price | shipping |
| = F  | > 20  | = 6      |
| = F  | ≤ 40  | = 6      |

$T_3$ :

(4)  $\text{CFD}^p \varphi_4 = \text{item}(\text{sale} \rightarrow \text{price}, T_4)$

|      |        |
|------|--------|
| sale | price  |
| = T  | ≥ 2.99 |
| = T  | < 9.99 |

$T_4$ :

Fig. 2. Example CFD's.

conjunction of all these conditions. Similarly,  $t[Y] \prec_{T_p}[Y]$  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 $^p$   $\varphi$ , denoted by  $I \models \varphi$ , if for each pair of tuples  $t_1, t_2$  in  $I$ , if  $t_1[X] = t_2[X] \prec_{T_p}[X]$ , then  $t_1[Y] = t_2[Y] \prec_{T_p}[Y]$ . 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  $\varphi$  is imposed only on the subset of tuples in  $I$  that match  $\text{LHS}(\varphi)$ , 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)  $t_1[Y] \prec_{T_p}[Y]$ , 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  $\Sigma$  of CFD $^p$ s, denoted by  $I \models \Sigma$ , if  $I \models \varphi$  for each CFD $^p$   $\varphi$  in  $\Sigma$ .

**Example 3.** The instance  $D_0$  of Fig. 1 satisfies  $\varphi_1$  and  $\varphi_2$  of Fig. 2, but neither  $\varphi_3$  nor  $\varphi_4$ . Indeed, tuple  $t_3$  violates (i.e., does not satisfy)  $\varphi_3$ , since  $t_3[\text{sale}] = \text{'F'}$  and  $20 < t_3[\text{price}] \leq 40$ , but  $t_3[\text{shipping}]$  is 20 instead of 6. Note that  $t_3$  matches  $\text{LHS}(\varphi_3)$  since it satisfies the condition specified by the conjunction of the pattern tuples in  $T_3$ . Similarly,  $t_1$  violates  $\varphi_4$ , since  $t_1[\text{sale}] = \text{'T'}$  but  $t_1[\text{price}] > 9.99$ . Observe that while it takes two tuples to violate a standard FD, a single tuple may violate a CFD $^p$ .

### 2.3 Special Cases

(1) A standard FD  $X \rightarrow Y$  [5] 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  $(X \rightarrow Y, T_p)$  [1] with  $T_p = \{t_{p1}, \dots, t_{pk}\}$  can be expressed as a set  $\{\varphi_1, \dots, \varphi_k\}$  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,  $\varphi_1$  and  $\varphi_2$  in Fig. 2 are CFD $^p$ s representing FD  $\text{cfd}_2$  and CFD  $\text{cfd}_3$  in Example 1, respectively. Note that all data quality rules in [7], [12] 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 [2] with built-in predicates ( $\neq, <, \leq, >, \geq$ ) in addition to equality (=). Consider two relational schemas  $R_1$  and  $R_2$ .

### 3.1 Syntax

A CIND $^p$   $\psi$  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  $\text{attr}(R_1)$  and

$\text{attr}(R_2)$ , respectively; (2)  $R_1[X] \subseteq R_2[Y]$  is a standard IND, referred to as the IND *embedded* in  $\psi$ ; and (3)  $T_p$  is a tableau, called the *pattern tableau* of  $\psi$  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  $\text{dom}(A)$ , or 'op a', where op is one of  $=, \neq, <, \leq, >, \geq$  and 'a' is a constant in  $\text{dom}(A)$ .

We denote  $X \cup X_p$  as  $\text{LHS}(\psi)$ ,  $Y \cup Y_p$  as  $\text{RHS}(\psi)$ , 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.** Fig. 3 shows two example CIND $^p$ s:  $\psi_1$  expresses the  $\text{pind}_1$  in Example 1, and  $\psi_2$  refines  $\psi_1$  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.,  $\psi_2$  assures that the sale tax rate in Delaware is 0.

### 3.2 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 $^p$   $\psi$ , denoted by  $(I_1, I_2) \models \psi$ , iff for each tuple  $t_1 \in I_1$ , if  $t_1[X_p] \prec_{T_p}[X_p]$ , then there exists a tuple  $t_2 \in I_2$  such that  $t_1[X] = t_2[Y]$  and  $t_2[Y_p] \prec_{T_p}[Y_p]$ .

That is, if  $t_1[X_p]$  matches the pattern tableau  $T_p[X_p]$ , then  $\psi$  assures the existence of  $t_2$  such that (1)  $t_1[X] = t_2[Y]$  as needed by the standard IND embedded in  $\psi$ ; and, moreover, (2)  $t_2[Y_p]$  must match the pattern tableau  $T_p[Y_p]$ . In other words,  $\psi$  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 $^p$   $\psi_1$ . Indeed, tuple  $t_1$  in item matches  $\text{LHS}(\psi_1)$  since  $t_1[\text{type}] \neq \text{'art'}$ , but there is no tuple  $t$  in tax such that  $t[\text{state}] = t_1[\text{state}] = \text{'WA'}$ . In contrast,  $D_0$  satisfies  $\psi_2$ .

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

### 3.3 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

(1)  $\psi_1 = (\text{item}[\text{state}; \text{type}] \subseteq \text{tax}[\text{state}; \text{nil}], T_1)$ ,

|                   |     |
|-------------------|-----|
| type              | nil |
| $\neq \text{art}$ |     |

(2)  $\psi_2 = (\text{item}[\text{state}; \text{type}, \text{state}] \subseteq \text{tax}[\text{state}; \text{rate}], T_2)$

|                   |       |      |
|-------------------|-------|------|
| type              | state | rate |
| $\neq \text{art}$ | = DL  | = 0  |

Fig. 3. Example CIND $^p$ s.

$B \in Y_p$ , 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.  $t_p[\text{price}] = 9.99$  and  $t'_p[\text{price}] \geq 19.99$ .

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

### 3.4 Special Cases

(1) A standard IND ( $R_1[X] \subseteq R_2[Y]$ ) can be expressed as a CIND<sup>p</sup> ( $R_1[X; \text{nil}] \subseteq R_2[Y; \text{nil}]$ ,  $T_p$ ) 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  $T_p = \{t_{p1}, \dots, t_{pk}\}$  can be expressed as a set  $\{\psi_1, \dots, \psi_k\}$  of CIND<sup>p</sup>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<sup>p</sup>S AND CIND<sup>p</sup>S

The satisfiability and implication problems are the two classical questions associated with any dependency languages [1], [2], [5]. In this section we investigate these problems for CFD<sup>p</sup>s and CIND<sup>p</sup>s, separately and taken together.

### 4.1 Satisfiability Analyses

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

The satisfiability analysis of conditional dependencies is not only of theoretical interest, but is also important in practice. Indeed, when CFD<sup>p</sup>s and CIND<sup>p</sup>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 [7], [12], [14].

#### 4.1.1 Satisfiability Analysis of CFD<sup>p</sup>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 [1], for a set  $\Sigma$  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<sup>p</sup>s, the same problem exists when it comes to CFD<sup>p</sup>s.

**Example 6.** Consider a CFD<sup>p</sup>  $\varphi = (R : A \rightarrow B, T_p)$  such that  $T_p = \{(- \parallel a), (- \not\parallel a)\}$ . There is no *nonempty* instance  $I$  of  $R$  that satisfies  $\varphi$ . Indeed, for any  $R$  tuple  $t$ ,  $\varphi$  requires that both  $t[B] = a$  and  $t[B] \neq a$ , which is impossible.

This problem is already NP-complete for CFDs [1]. Below we show that it remains the same complexity for CFD<sup>p</sup>s despite their increased expressive power.

**Proposition 1.** *The satisfiability problem for CFD<sup>p</sup>s is NP-complete.*

**Proof.** The lower bound follows from the NP-hardness of their CFDs counterparts [1], since CFDs are a special case of CFD<sup>p</sup>s. The upper bound is verified by presenting an

NP algorithm that, given a set  $\Sigma$  of CFD<sup>p</sup>s defined on a relational schema  $R$ , determines whether  $\Sigma$  is satisfiable.

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

Assume *w.l.o.g.* that the attributes  $\text{attr}(R) = \{A_1, \dots, A_m\}$  and the total number of pattern tuples in all pattern tableaux  $T_p$  of CFD<sup>p</sup>s in  $\Sigma$  is  $h$ . For each  $i \in [1, m]$ , define the active domain of  $A_i$  to be a set  $\text{adom}(A_i) = C_0 \cup C_1$ , where (1)  $C_0$  consists of all constants in  $T_p[A_i]$  of all pattern tableaux  $T_p$  in  $\Sigma$ , and if  $C_0$  is empty, we further let  $C_0 = \{a_1, a_2\}$ , where  $a_1, a_2 \in \text{dom}(A_i)$  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  $C_0 = \{a_1, \dots, a_k\}$  ( $k \geq 1$ );
- 2) Add a constant  $b_{01} \in \text{dom}(A_i)$  to  $C_1$  such that  $b_{01} < a_1$  if there exists one; And also add another constant  $b_{02} \in \text{dom}(A_i)$  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  $b_{j1} \in \text{dom}(A_i)$  to  $C_1$  such that  $a_j < b_{j1} < a_{j+1}$  if there exists one; And also add another constant  $b_{j2} \in \text{dom}(A_i)$  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  $b_{k1} \in \text{dom}(A_i)$  to  $C_1$  such that  $b_{k1} > a_k$  if there exists one; And also add another constant  $b_{k2} \in \text{dom}(A_i)$  to  $C_1$  such that  $b_{k2} > a_k$  and  $b_{k2} \neq b_{k1}$  if there exists one.

Moreover, the number of elements in  $\text{adom}(A_i)$  is at most  $3 * h + 2$ . Then one can easily verify that  $\Sigma$  is satisfiable iff there exists a mapping  $\rho$  from  $t[A_i]$  to  $\text{adom}(A_i)$  ( $i \in [1, m]$ ) such that  $I = \{(\rho(t[A_1]), \dots, \rho(t[A_m]))\}$  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  $t[A_i] \in \text{adom}(A_i)$  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  $\Sigma$ . Hence the algorithm is in NP, and so is the problem.  $\square$

It is known [1] 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<sup>p</sup>s. This tells us that the increased expressive power of CFD<sup>p</sup>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 [1], the result below is new.

**Theorem 2.** *In the absence of finite domain attributes, the satisfiability problem for CFD<sup>p</sup>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. [33]).

We next show the reduction from the 3SAT problem. Consider an instance  $\phi = C_1 \wedge \dots \wedge C_n$  of 3SAT, where all the variables in  $\phi$  are  $x_1, \dots, x_m$ ,  $C_j$  is of the form  $y_{j1} \vee y_{j2} \vee y_{j3}$  such that for each  $i \in [1, 3]$ ,  $y_{ji}$  is either  $x_{p_{ji}}$  or  $\overline{x_{p_{ji}}}$  for  $p_{ji} \in [1, m]$ . Given  $\phi$ , we construct a relational schema  $R$  and a set  $\Sigma$  of CFD<sup>s</sup> defined on  $R$  such that  $\phi$  is satisfiable iff  $\Sigma$  is satisfiable.

- 1) We first define the relational schema  $R(X_1, \dots, X_m, C_1, \dots, C_n, Z)$ , where all attributes share a common infinite domain  $\text{dom}$  that contains constant  $a$ . Intuitively, for each  $R$  tuple  $t$ ,  $t[X_1, \dots, X_m]$  specifies a truth assignment  $\xi$  for variables  $x_1, \dots, x_m$  of  $\phi$ , and  $t[C_i]$  ( $i \in [1, n]$ ) and  $t[Z]$  are the truth values of clause  $C_i$  and sentence  $\phi$  w.r.t. the assignment  $\xi$ , respectively.
- 2) We then construct the set CFD<sup>s</sup>  $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \dots \cup \Sigma_n \cup \Sigma_{n+1}$ , defined as follows.
  - a)  $\Sigma_0$  contains  $n + 1$  CFD<sup>s</sup>, which intuitively encode the relationships of the truth values between the clauses  $C_1, \dots, C_n$  and sentence  $\phi$ .  
For each clause  $C_i$  ( $i \in [1, n]$ ), we add to  $\Sigma_0$  a CFD<sup>s</sup>  $\varphi_i = (C_1, \dots, C_n \rightarrow Z, T_{pi})$ , in which  $T_{pi} = \{t_{pi}\}$  such that  $t_{pi}[C_i, Z] = (\neq a \parallel \neq a)$  and  $t_{pi}[C_j] = \text{'_'} for any  $j \neq i$  and  $j \in [1, n]$ . We also add to  $\Sigma_0$  a CFD<sup>s</sup>  $\varphi_0 = (C_1, \dots, C_n \rightarrow Z, T_{p0})$ , where  $T_{p0} = \{(= a, \dots, = a \parallel = a)\}$ . Intuitively, we use  $\neq a$  and  $= a$  to represent *false* and *true*, respectively.$
  - b) For  $i \in [1, n]$ ,  $\Sigma_i$  contains 8 CFD<sup>s</sup>, which intuitively encode the relationships of the truth values between the clause  $C_i$  and its three variables.  
For clause  $C_i = x_{j1} \vee \overline{x_{j2}} \vee x_{j3}$  of  $\phi$  with  $1 \leq j_1, j_2, j_3 \leq m$ , we define CFD<sup>s</sup> (a)  $\varphi_{i,0} = (X_{j1}, X_{j2}, X_{j3} \rightarrow C_i, T_{pi,0})$  with  $T_{pi,0} = \{(< a, < a, < a \parallel = a)\}$ , (b)  $\varphi_{i,1} = (X_{j1}, X_{j2}, X_{j3} \rightarrow C_i, T_{pi,2})$  with  $T_{pi,1} = \{(< a, < a, \geq a \parallel = a)\}$ , and (c)  $\varphi_{i,2} = (X_{j1}, X_{j2}, X_{j3} \rightarrow C_i, T_{pi,2})$  with  $T_{pi,2} = \{(< a, \geq a, < a \parallel \neq a)\}$ . Similarly, we can define the rest five CFD<sup>s</sup>  $\varphi_{i,3}, \varphi_{i,4}, \varphi_{i,5}, \varphi_{i,6}$  and  $\varphi_{i,7}$ . Intuitively, we further use  $< a$  and  $\geq a$  to represent *false* and *true* for a variable, respectively.
  - c)  $\Sigma_{n+1}$  contains a single CFD<sup>s</sup>  $\varphi_{n+1} = (Z \rightarrow Z, T_{p(n+1)})$  with  $T_{p(n+1)} = \{(- \parallel = a)\}$ . Intuitively,  $\varphi_{n+1}$  requires that for any  $R$  tuple  $t$ ,  $t[Z] = a$ .

Observe that  $\Sigma$  contains  $8 * m + n + 2$  CFD<sup>s</sup> in total. Thus the reduction is in PTIME.

We now show that  $\phi$  is satisfiable iff  $\Sigma$  is satisfiable.

Assume first that  $\Sigma$  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)  $\Sigma_{n+1}$  forces  $t[Z] = a$ , (b)  $\Sigma_0$  forces  $t[C_1, \dots, C_n] = (a, \dots, a)$ , and (c) for each clause  $C_i$  ( $i \in [1, n]$ ) with variables  $X_{j1}, X_{j2}, X_{j3}$ ,  $\Sigma_j$  forces that  $t[X_{j1}, X_{j2}, X_{j3}]$  does not match the LHS of the CFD<sup>s</sup> that force  $t[C_i] \neq a$ . From the tuple  $t$ , we can construct a truth

assignment  $\xi$  of  $\phi$  such that  $\xi(x_i) = \text{false}$  if  $t[X_i] < a$  and  $\xi(x_i) = \text{true}$  if  $t[X_i] \geq a$  ( $i \in [1, m]$ ). Since  $\{t\} \models \Sigma$ , it is easy to verify that the truth assignment  $\xi$  makes  $\phi$  true.

Conversely, if  $\phi$  is satisfiable, there exists a truth assignment  $\xi$  that makes  $\phi$  true. We construct a tuple  $t$  of  $R$  as follows: (a)  $t[C_1, \dots, C_n, Z] = (a, \dots, a)$  and (b) for each  $i \in [1, m]$ ,  $t[X_i] = a_i$  such that (a)  $a_i \in \text{dom}$  and  $a_i \geq a$  if  $\xi(x_i) = \text{true}$  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.  $\square$

#### 4.1.2 Satisfiability Analysis of CIND<sup>s</sup>

Like FDs, one can specify arbitrary INDS or CINDs without worrying about their satisfiability. Below we show that CIND<sup>s</sup> preserve this nice property, by extending the proof of its counterpart in [2].

**Proposition 3.** Any set of CIND<sup>s</sup> is always satisfiable.

**Proof.** Given a set  $\Sigma$  of CIND<sup>s</sup> over a database schema  $\mathcal{R}(R_1, \dots, R_n)$ , 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  $\text{adom}(A)$ , which consists of certain data values in  $\text{dom}(A)$ . Second, using these active domains, we construct  $D$ .

- 1) We start with the construction of active domains.
  - (a) For each attribute  $A$ , initialize  $\text{adom}(A)$  along the same lines as the one for CFD<sup>s</sup> in Proposition 1;
  - (b) For each CIND<sup>s</sup>  $(R_a[A_1, A_2, \dots, A_m; X_p] \subseteq R_b[B_1, \dots, B_m; Y_p], T_p)$  in  $\Sigma$ , let  $\text{adom}(B_i) = \text{adom}(B_i) \cup \text{adom}(A_i)$  for each  $i \in [1, m]$ , and this rule is repeatedly applied until a fixpoint of  $\text{adom}(A)$  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  $R_i(A_1, \dots, A_k) \in \mathcal{R}$ , we define  $I_i = \text{adom}(A_1) \times \dots \times \text{adom}(A_k)$ , where  $\times$  is the Cartesian Product operation [5]. Let  $D = \{I_1, \dots, I_n\}$ , then it is easy to verify that  $D$  is nonempty and  $D \models \Sigma$ .  $\square$

#### 4.1.3 Satisfiability Analysis of CFD<sup>s</sup> and CIND<sup>s</sup>

The satisfiability problem for CFDs and CINDs taken together is undecidable [2]. Since CFD<sup>s</sup> and CIND<sup>s</sup> subsume CFDs and CINDs, respectively, we immediately have the following.

**Corollary 4.** The satisfiability problem for CFD<sup>s</sup> and CIND<sup>s</sup> is undecidable.

## 4.2 Implication Analyses

The implication problem is to determine, given a set  $\Sigma$  of dependencies and another dependency  $\phi$ , whether or not  $\Sigma$  entails  $\phi$ , 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 [1], [2].



**Example 7.** The CFD<sup>p</sup>s in Fig. 2 imply another CFD<sup>p</sup>  $\varphi = \text{item}(\text{sale}, \text{price} \rightarrow \text{shipping}, T)$ , where  $T$  consists of a single pattern tuple ( $\text{sale} = 'F', \text{price} = 30 \parallel \text{shipping} = 6$ ). Thus in the presence of the CFD<sup>p</sup>s in Fig. 2,  $\varphi$  is redundant.

#### 4.2.1 Implication Analysis of CFD<sup>p</sup>s

We first show that the implication problem for CFD<sup>p</sup>s retains the same complexity as their CFDs counterpart, verified by extending the proof of its counterpart in [1].

**Proposition 5.** *The implication problem for CFD<sup>p</sup>s is conNP-complete.*

**Proof.** The lower bound follows from the conNP-hardness of their CFDs counterpart [1], since CFDs are a special case of CFD<sup>p</sup>s. The conNP 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  $t_1[X] = t_2[X] \succ T_p[X]$ , but either  $t_1[Y] \neq t_2[Y]$  or  $t_1[Y] \not\prec T_p[Y]$  (resp.  $t_2[Y] \not\prec T_p[Y]$ ). Thus it suffices to consider instances  $I$  with two tuples only for deciding whether  $\Sigma \not\models \varphi$ .

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

Based on these, we give an NP algorithm as follows: (1) Guess two  $R$  tuples  $t_1$  and  $t_2$  such that  $t_1[A_i], t_2[A_i] \in \text{adom}(A_i)$  for each  $i \in [1, m]$ . (2) Check whether  $I = \{t_1, t_2\}$  satisfies  $\Sigma$ , but not  $\varphi$ . 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  $\Sigma$ . Hence the algorithm is in NP, and so is the problem.  $\square$

Similar to the satisfiability analysis, it is known [1] 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<sup>p</sup>s, which does not find a counterpart in [1].

**Proposition 6.** *In the absence of finite domain attributes, the implication problem for CFD<sup>p</sup>s is conNP-complete.*

**Proof.** The problem is in conNP by Proposition 5. The conNP-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<sup>p</sup>s, where 3SAT is NP-complete (cf. Proposition 2). Given an instance  $\phi$  of 3SAT, we construct a relational schema  $R$  and a set  $\Sigma \cup \{\varphi\}$  of CFD<sup>p</sup>s defined on  $R$  such that  $\phi$  is satisfiable iff  $\Sigma \not\models \varphi$ .

The relational schema  $R$  and the set  $\Sigma$  of CFD<sup>p</sup>s are the same as the corresponding ones in Proposition 2. Moreover,  $\varphi$  is defined as  $(Z \rightarrow Z, T_p)$ , where  $T_p = \{(- \parallel \neq a)\}$ .

Intuitively,  $\varphi$  requires that for any  $R$  tuple  $t$ ,  $t[Z] \neq a$ . Along the same lines as Proposition 2, one can easily verify that  $\phi$  is satisfiable iff  $\Sigma \not\models \varphi$ . Thus the problem is conNP-hard.  $\square$

#### 4.2.2 Implication Analysis of CIND<sup>p</sup>s

We next show that CIND<sup>p</sup>s do not make their implication analysis harder, verified by extending the proof of their CINDs counterpart given in [2].

**Proposition 7.** *The implication problem for CIND<sup>p</sup>s is EXPTIME-complete.*

**Proof.** The implication problem for CINDs is EXPTIME-hard [2]. Since CIND<sup>p</sup>s subsume CINDs, the lower bound carries over to CIND<sup>p</sup>s immediately. The EXPTIME upper bound is shown by presenting an EXPTIME algorithm that, given a set  $\Sigma \cup \{\psi\}$  of CIND<sup>p</sup>s over a database schema  $\mathcal{R}$ , determines whether  $\Sigma \models \psi$  or not.

We next present the EXPTIME algorithm. Consider  $\mathcal{R} = (R_1, \dots, R_n)$  and  $\psi = (R_a[X; X_p] \subseteq R_b[Y; Y_p], T_p)$ . And for each attribute  $A$ , define the active domain  $\text{adom}(A)$  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  $t_i[A] \in \text{adom}(A)$  for each attribute  $A \in \text{attr}(R_i)$ . There is an edge  $e = ('R_i : t_i', 'R_j : t_j')$  in  $E$  iff there exists a CIND<sup>p</sup>  $\phi = (R_i[U; U_p] \subseteq R_j[V; V_p], T_{p_\phi})$  in  $\Sigma$  such that  $t_i[U_p] \succ T_{p_\phi}[U_p]$ ,  $t_j[V] = t_i[U]$  and  $t_j[V_p] \succ T_{p_\phi}[V_p]$ , and  $e$  is labeled with the CIND<sup>p</sup>  $\phi$ , 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  $t_a[X_p] \succ T_p[X_p]$ , and  $S_b$  be the set of nodes ' $R_b : t_b$ ' such that  $t_b[Y_p] \succ T_p[Y_p]$ , 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

$\psi$ , by collecting those tuples attached on the end nodes of edges whose labels become empty at step 4; and (b) if the algorithm returns ‘yes’, there exist no instances  $D$  such that  $D \models \Sigma$ , but not  $\psi$ .

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  $\Sigma$  and  $\psi$ , 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 [34].

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

It is known [2] 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<sup>p</sup>s.

**Theorem 8.** *In the absence of finite domain attributes, the implication problem for CIND<sup>p</sup>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 [2].

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  $\mathcal{R} (R_1, \dots, R_n)$ , we construct another database schema  $\mathcal{R}' (R'_1, \dots, R'_n)$ , in which each relation  $R'_i$  ( $i \in [1, n]$ ) consists of infinite domain attributes only, and a set  $\Sigma' \cup \{\psi'\}$  of CIND<sup>p</sup>s on  $\mathcal{R}'$  such that  $\Sigma \models \psi$  iff  $\Sigma' \models \psi'$ .

- 1) We start with constructing  $\mathcal{R}'$ . For each  $R_i(A_1, \dots, A_k)$  of  $\mathcal{R}$ , we define  $R'_i(A'_1, \dots, A'_k)$  such that for each attribute  $A'_j$  ( $j \in [1, k]$ ), let  $\text{dom}(A'_j) = \text{dom}(A_j)$  if  $\text{dom}(A_j)$  is infinite, and let  $\text{dom}(A'_j)$  be integer, a totally ordered infinite domain, if  $\text{dom}(A_j)$  is finite. Moreover, we define a mapping  $\rho_{i,j}$  for each finite domain  $\text{dom}(A_j) = \{a_1, \dots, a_h\}$  to integer: (a) Randomly choose  $h$  consecutive integers  $\{b_1, \dots, b_h\}$  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  $\rho$  when it is clear from the context.
- 2) We next define  $\Sigma'$  and  $\psi'$  on  $\mathcal{R}'$  based on the mappings defined above. For each CIND  $(R_a[X; A_1, \dots, A_{m1}] \subseteq R_b[Y; B_1, \dots, B_{m2}], T_p)$  in  $\Sigma$  and each  $t_p \in T_p$ , we define another CIND<sup>p</sup>  $(R'_a[X'; A'_1, \dots, A'_{m1}, X'_p] \subseteq R'_b[Y'; B'_1, \dots, B'_{m2}, Y'_p], T'_p)$ , where (a)  $X'$  (resp.  $Y'$ ,  $A'_1, \dots, A'_{m1}$  and  $B'_1, \dots, B'_{m2}$ ) corresponds to  $X$  (resp.  $Y$ ,  $A_1, \dots, A_{m1}$  and

$B_1, \dots, B_{m2}$ ); (b)  $X'_p$  (resp.  $Y'_p$ ) corresponds to those finite domain attributes in  $R_a$  (resp.  $R_b$ ), but not in  $A_1, \dots, A_{m1}$  (resp.  $B_1, \dots, B_{m2}$ ); and (c)  $T'_p = \{t'_{p1}, t'_{p2}, t'_{p3}\}$  such that for each attribute  $A'$  in  $A'_1, \dots, A'_{m1}$  or  $B'_1, \dots, B'_{m2}$ , (i)  $t'_{p1}[A'] = t_p[A]$  and  $t'_{p2}[A'] = t'_{p3}[A'] = \text{'_'} if  $\text{dom}(A)$  is infinite, and (ii)  $t'_{p1}[A'] = \rho(t_p[A])$  and  $t'_{p2}[A'] = t'_{p3}[A'] = \text{'_'} if  $\text{dom}(A)$  is finite; and (iii) for the rest attributes  $A'$  in  $X'_p$  or  $Y'_p$ ,  $t'_{p1}[A'] = \text{'_'} , t'_{p2}[A'] = \text{'> } \rho.b_0'$ , and  $t'_{p3}[A'] = \text{'< } \rho.b_{h+1}'$ . Similarly, we can define a set  $\Sigma_\psi$  of CIND<sup>p</sup>s from  $\psi$  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<sup>p</sup>  $\psi' \in \Sigma_\psi$ . Following from this, the problem is EXPTIME-hard.  $\square$

#### 4.2.3 Implication Analysis of CFD<sup>p</sup>s and CIND<sup>p</sup>s

When CFD<sup>p</sup>s and CIND<sup>p</sup>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 [5]. Since CFD<sup>p</sup>s and CIND<sup>p</sup>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<sup>p</sup>s and CIND<sup>p</sup>s is undecidable.*

#### 4.2.4 Remarks

Inference systems play an important role for the implication analyses [5]. For the inference system of CFD<sup>p</sup>s and CIND<sup>p</sup>s alone, we can readily extend the one for CFDs [1] and CINDs [2], 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<sup>p</sup>s and CIND<sup>p</sup>s together is not finitely axiomatizable by Corollary 9.

#### 4.2.5 Summary

The complexity bounds for reasoning about CFD<sup>p</sup>s and CIND<sup>p</sup>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 [1], [2]. The results tell us the following.

- 1) Despite the increased expressive power, CFD<sup>p</sup>s and CIND<sup>p</sup>s do not complicate the static analyses in the general case: the satisfiability and implication problems for CFD<sup>p</sup>s and CIND<sup>p</sup>s have the same complexity bounds as their counterparts for CFDs and CINDs, taken separately or together.
- 2) In the special case when CFD<sup>p</sup>s and CIND<sup>p</sup>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<sup>p</sup>s and CIND<sup>p</sup>s comes at a price in this special case.

### 5 VALIDATION OF CFD<sup>p</sup>s AND CIND<sup>p</sup>s

If CFD<sup>p</sup>s and CIND<sup>p</sup>s are to be used as data quality rules, the first question we have to settle is how to effectively detect



TABLE 1  
Summary of Complexity Results

| $\Sigma$                                 | General setting |                  | Infinite domain only |                  |
|--|-----------------|------------------|----------------------|------------------|
|  | Satisfiability  | Implication      | Satisfiability       | Implication      |
| CFDs [1]                                 | NP-complete     | coNP-complete    | PTIME                | PTIME            |
| CFD <sup>p</sup> s                       | NP-complete     | coNP-complete    | NP-complete          | coNP-complete    |
| CINDs [2]                                | $O(1)$          | EXPTIME-complete | $O(1)$               | PSPACE-complete  |
| CIND <sup>p</sup> s                      | $O(1)$          | EXPTIME-complete | $O(1)$               | EXPTIME-complete |
| CFDs + CINDs [2]                         | undecidable     | undecidable      | undecidable          | undecidable      |
| CFD <sup>p</sup> s + CIND <sup>p</sup> s | undecidable     | undecidable      | undecidable          | undecidable      |

errors and inconsistencies as violations of these dependencies, by leveraging functionality supported by commercial DBMS. More specifically, consider a database schema  $\mathcal{R} = (R_1, \dots, R_n)$ , 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  $\Sigma$  of CFD<sup>p</sup>s and CIND<sup>p</sup>s defined on  $\mathcal{R}$ , and a database instance  $D = (I_1, \dots, I_n)$  of  $\mathcal{R}$  as input, the subset  $(I'_1, \dots, I'_n)$  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<sup>p</sup> or CIND<sup>p</sup> in  $\Sigma$ . We denote the set as  $\text{vio}(D, \Sigma)$ , referred to it as *the violation set of  $D$  w.r.t.  $\Sigma$* .

In this section we develop SQL-based techniques for error detection based on CFD<sup>p</sup>s and CIND<sup>p</sup>s. The main result of the section is as follows.

**Theorem 10.** *Given a set  $\Sigma$  of CFD<sup>p</sup>s and CIND<sup>p</sup>s defined on  $\mathcal{R} = (R_1, \dots, R_n)$  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  $\text{vio}(D, \Sigma)$ , 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  $\Sigma$ .*

Let  $\Sigma_{\text{cfdp}}^i$  be the set of all CFD<sup>p</sup>s in  $\Sigma$  defined on the same relational schema  $R_i$ , and  $\Sigma_{\text{cindp}}^{(i,j)}$  the set of all CIND<sup>p</sup>s in  $\Sigma$  from  $R_i$  to  $R_j$ , for  $i, j \in [1, n]$ . We show the following. (a) The violation set  $\text{vio}(D, \Sigma_{\text{cfdp}}^i)$  can be computed by *two* SQL queries. (b) Similarly,  $\text{vio}(D, \Sigma_{\text{cindp}}^{(i,j)})$  can be computed by a *single* SQL query. (c) These SQL queries use pattern tableaux of CFD<sup>p</sup>s (CIND<sup>p</sup>s) encoded with data tables, and hence their sizes are independent of  $\Sigma$ . 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<sup>p</sup>s and CIND<sup>p</sup>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<sup>p</sup>s or CIND<sup>p</sup>s.

### 5.1 Encoding CFD<sup>p</sup>s and CIND<sup>p</sup>s with Data Tables

We first show the following, by extending the encoding of [1], [6]. The pattern tableaux of all CFD<sup>p</sup>s in  $\Sigma_{\text{cfdp}}^i$  can be encoded with *three data tables*, and the pattern tableaux of all CIND<sup>p</sup>s in  $\Sigma_{\text{cindp}}^{(i,j)}$  can be represented as *four data tables*, no matter how many dependencies are in the sets.

#### 5.1.1 Encoding CFD<sup>p</sup>s

We encode all pattern tableaux in  $\Sigma_{\text{cfdp}}^i$  with three tables  $\text{enc}_L$ ,  $\text{enc}_R$  and  $\text{enc}_{\neq}$ , where  $\text{enc}_L$  (resp.  $\text{enc}_R$ ) encodes the

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

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

We populate  $\text{enc}_L$ ,  $\text{enc}_R$  and  $\text{enc}_{\neq}$  as follows. For each CFD<sup>p</sup>  $\varphi = R(X \rightarrow Y, T_p)$  in  $\Sigma_{\text{cfdp}}^i$ , we generate a distinct  $\text{cid}_{\varphi}$  for it, and do the following.

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

**Example 8.** Recall from Fig. 2 CFD<sup>p</sup>s  $\varphi_2$ ,  $\varphi_3$  and  $\varphi_4$  defined on relation item. The three CFD<sup>p</sup>s are encoded with the tables shown in Fig. 4: (a)  $\text{enc}_L$  consists of attributes:  $\text{cid}$ ,  $\text{sale}$ ,  $\text{price}$ ,  $\text{price}_{>}$  and  $\text{price}_{\geq}$ ; (b)  $\text{enc}_R$  consists of  $\text{cid}$ ,  $\text{shipping}$ ,  $\text{price}$ ,  $\text{price}_{\geq}$  and  $\text{price}_{<}$ ; those attributes in a table with only 'null' pattern values do not

| (1) $\text{enc}_L$ |      |       |                       |                       | (2) $\text{enc}_R$ |          |       |                    |                       | (3) $\text{enc}_{\neq}$ |     |     |     |
|--------------------|------|-------|-----------------------|-----------------------|--------------------|----------|-------|--------------------|-----------------------|-------------------------|-----|-----|-----|
| cid                | sale | price | price <sub>&gt;</sub> | price <sub>&lt;</sub> | cid                | shipping | price | price <sub>≥</sub> | price <sub>&lt;</sub> | cid                     | pos | att | val |
| 2                  | T    | null  | null                  | null                  | 2                  | 0        | null  | null               | null                  |                         |     |     |     |
| 3                  | F    | —     | 20                    | 40                    | 3                  | 6        | null  | null               | null                  |                         |     |     |     |
| 4                  | T    | null  | null                  | null                  | 4                  | null     | —     | 2.99               | 9.99                  |                         |     |     |     |

Fig. 4. Encoding example of CFD<sup>p</sup>s.

contribute to error detection, and are thus omitted; And (c)  $\text{enc}_{\neq}$  is empty since all these CFD<sup>p</sup>s have no negation patterns. One can easily reconstruct these CFD<sup>p</sup>s from tables  $\text{enc}_L$ ,  $\text{enc}_R$  and  $\text{enc}_{\neq}$  by collating the tuples based on cid.

### 5.1.2 Encoding CIND<sup>p</sup>s

All CIND<sup>p</sup>s in  $\Sigma_{\text{cindp}}^{(i,j)}$  can be encoded with four tables  $\text{enc}$ ,  $\text{enc}_L$ ,  $\text{enc}_R$  and  $\text{enc}_{\neq}$ . Here  $\text{enc}_L$  (resp.  $\text{enc}_R$ ) and  $\text{enc}_{\neq}$  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<sup>p</sup>s. We use  $\text{enc}$  to encode the INDS embedded in CIND<sup>p</sup>s, which consists of the following attributes: (1) cid representing the id of a CIND<sup>p</sup>, and (2) those  $X$  attributes of  $R_i$  and  $Y$  attributes of  $R_j$  appearing in some CIND<sup>p</sup>s in  $\Sigma_{\text{cindp}}^{(i,j)}$ . Note that the number of attributes in  $\text{enc}$  is bounded by  $|R_i| + |R_j| + 1$ , where  $|R_i|$  is the arity of  $R_i$ .

For each CIND<sup>p</sup>  $\psi = (R_i[A_1 \dots A_m; X_p] \subseteq R_j[B_1 \dots B_m; Y_p], T_p)$  in  $\Sigma_{\text{cindp}}^{(i,j)}$ , we generate a distinct cid  $\text{id}_{\psi}$  for it, and do the following.

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

**Example 9.** Fig. 5 shows the coding of CIND<sup>p</sup>s  $\psi_1$  and  $\psi_2$  given in Fig. 3. We use  $\text{state}_L$  and  $\text{state}_R$  in  $\text{enc}$  to denote the occurrences of attribute *state* in *item* and *tax*, respectively. In  $\text{enc}_L$  and  $\text{enc}_R$ , the attributes with only ‘null’ patterns are omitted, for the same reason as CFD<sup>p</sup>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  $\Sigma$ , regardless of the size of  $\Sigma$ . 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  $\text{vio}(D, \Sigma_{\text{cfdp}}^i)$ . 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  $\text{vio}(D, \Sigma_{\text{cindp}}^{(i,j)})$ . Putting these query answers together, we get  $\text{vio}(D, \Sigma)$ , the violation set of  $D$  w.r.t.  $\Sigma$ .

### 5.2.1 SQL Queries for CIND<sup>p</sup>s

Below we show how the SQL query  $Q_{(i,j)}$  is generated for validating CIND<sup>p</sup>s in  $\Sigma_{\text{cindp}}^{(i,j)}$ , which has not been studied by previous work. For the lack of space, we put the generation of detection queries for CFD<sup>p</sup>s in the supplementary material, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TKDE.2015.2451632>, which is an extension of the SQL techniques for CFDs and eCFDs discussed in [1] and [6], respectively.

The query  $Q_{(i,j)}$  for the validation of  $\Sigma_{\text{cindp}}^{(i,j)}$  is given as follows, which capitalizes on the data tables  $\text{enc}$ ,  $\text{enc}_L$ ,  $\text{enc}_R$  and  $\text{enc}_{\neq}$  that encode CIND<sup>p</sup>s in  $\Sigma_{\text{cindp}}^{(i,j)}$ .

```

select  $R_i.*$  from  $R_i, \text{enc}_L L, \text{enc}_{\neq} N$ 
where  $R_i.X \prec L$  and  $R_i.X \prec N$  and not exists (
  select  $R_j.*$  from  $R_j, \text{enc}_H H, \text{enc}_R R, \text{enc}_{\neq} N$ 
  where  $R_i.X = R_j.Y$  and  $L.\text{cid} = R.\text{cid}$  and
     $L.\text{cid} = H.\text{cid}$  and  $R_j.Y \prec R$  and  $R_j.Y \prec N$ )

```

Here (1)  $X = \{A_1, \dots, A_{m1}\}$  and  $Y = \{B_1, \dots, B_{m2}\}$  are the sets of attributes of  $R_i$  and  $R_j$  appearing in  $\Sigma_{\text{cindp}}^{(i,j)}$ , respectively; (2)  $R_i.X \prec L$  is the conjunction of

```

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

```

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

```

not exists (select * from  $N$ 
  where  $L.\text{cid} = N.\text{cid}$  and  $N.\text{pos} = \text{'LHS'}$  and
     $N.\text{att} = \text{'A}_k'$  and  $R_i.A_k = N.\text{val}$ );

```

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

Intuitively, (1)  $R_i.X \prec L$  and  $R_i.X \prec N$  ensure that the  $R_i$  tuples selected match the LHS patterns of some CIND<sup>p</sup>s in  $\Sigma_{\text{cindp}}^{(i,j)}$ ; (2)  $R_j.Y \prec R$  and  $R_j.Y \prec N$  check the corresponding RHS patterns of these CIND<sup>p</sup>s on  $R_j$  tuples; (3)  $R_i.X = R_j.Y$

| (1) $\text{enc}$ |                    |                    | (2) $\text{enc}_L$ |      |       | (3) $\text{enc}_R$ |      | (4) $\text{enc}_{\neq}$ |     |      |     |
|------------------|--------------------|--------------------|--------------------|------|-------|--------------------|------|-------------------------|-----|------|-----|
| cid              | state <sub>L</sub> | state <sub>R</sub> | cid                | type | state | cid                | rate | cid                     | pos | att  | val |
| 1                | 1                  | 1                  | 1                  | —    | null  | 1                  | null | 1                       | LHS | type | art |
| 2                | 1                  | 1                  | 2                  | —    | DL    | 2                  | 0    | 2                       | LHS | type | art |

Fig. 5. Encoding example of CIND<sup>p</sup>s.

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  $\Sigma_{CIND^p}^{(i,j)}$ .

**Example 10.** Using the coding of Fig. 5, an SQL query  $Q$  for checking  $CIND^p$ s  $\psi_1$  and  $\psi_2$  of Fig. 3 is given as follows:

```
select  $R_1.*$  from item  $R_1$ ,  $enc_L L$ ,  $enc_{\neq} N$ 
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  $N.pos = \text{'LHS'}$  and
     $N.att = \text{'type'}$  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  $N.pos = \text{'LHS'}$  and
     $N.att = \text{'state'}$  and  $R_1.state = N.val$ ) and
not exist (select  $R_2.*$  from tax  $R_2$ ,  $enc_H H$ ,  $enc_R R$ 
  where ( $H.state_L$  is null or  $H.state_R$  is null or
     $H.state_L \neq H.state_R$  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  $N.pos = \text{'RHS'}$ 
    and  $N.att = \text{'rate'}$  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  $enc_{\neq}$ , 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 versus their counterparts  $CFDs$  and  $CINDs$ , separately and taken together.

### 6.1 Experimental Settings

We first present our experimental settings.

#### 6.1.1 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 US Department of Health & Human Services [35]. 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 six  $CFD^p$ s and three  $CIND^p$ s for HOSP, shown below in an informal way for easy of understanding.

```
 $\phi_1$ : hcahps (zip =  $\_$  and city =  $\_$   $\rightarrow$  state =  $\_$ )
 $\phi_2$ : hcahps (hid =  $\_$   $\rightarrow$  hname =  $\_$  and county =  $\_$  and
  addr =  $\_$  and phn =  $\_$ )
 $\phi_3$ : hcahps (hid =  $\_$   $\rightarrow$  msrrp =  $\_$ )
 $\phi_4$ : hcahps (mid =  $\_$   $\rightarrow$  mq =  $\_$  and mad =  $\_$ )
 $\phi_5$ : hcahps (hid =  $\_$  and mncs = 'Not Available'  $\rightarrow$  mfn
   $\geq 1$  and mfn  $\leq 14$ )
 $\phi_6$ : hcahps (hid =  $\_$  and mid =  $\_$  and mncs  $\neq$  'Not
  Available' and mncs  $\neq$  'Fewer than 100'  $\rightarrow$  map  $\geq 0$ 
  and map  $\leq 100$ )
 $\psi_1$ : hcahps (mid, mq; nil)  $\subseteq$  hcahps-state (mid, mq; nil)
 $\psi_2$ : hcahps (mid, state; nil)  $\subseteq$  hcahps-state (mid, state; nil)
 $\psi_3$ : hcahps (mid, state; mncs  $\neq$  'Not Available')  $\subseteq$ 
  hcahps-state (mid, state; map  $\geq 0$  and map  $\leq 100$ )
```

For comparison, we also designed the  $CFDs$  and  $CINDs$  counterparts of the above  $CFD^p$ s and  $CIND^p$ s. Here  $\phi_1$ - $\phi_4$  and  $\psi_1$ - $\psi_2$  are indeed  $CFDs$  and  $CINDs$ , respectively, while  $\phi_5$ ,  $\phi_6$  and  $\psi_3$  are not. We hence further designed  $\phi'_5$ ,  $\phi'_6$  and  $\psi'_3$  to approximate  $\phi_5$ ,  $\phi_6$  and  $\psi_3$ , respectively.

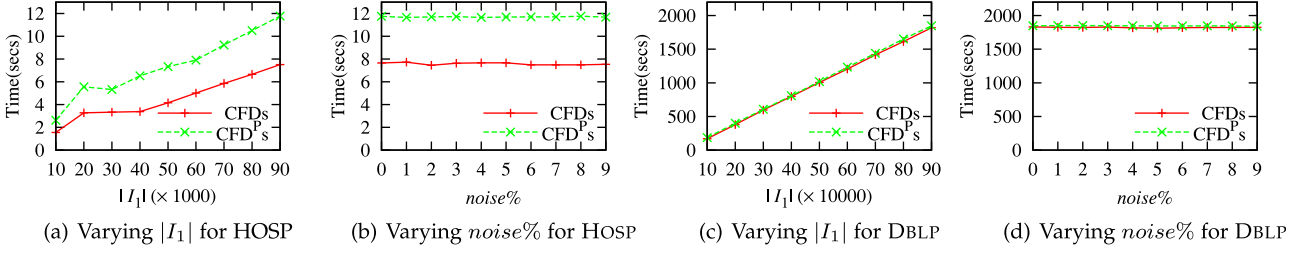
```
 $\phi'_5$ : hcahps (hid =  $\_$  and mncs = 'Not Available'  $\rightarrow$  mfn
  =  $\_$ )
 $\phi'_6$ : hcahps (hid =  $\_$  and mid =  $\_$  and mncs =  $\_$   $\rightarrow$  map
  =  $\_$ )
 $\psi'_3$ : hcahps (mid, state; mncs =  $\_$ )  $\subseteq$  hcahps-state (mid,
  state; map =  $\_$ )
```

(2) DBLP is a repository of computer science publications from 1946 to 2014 [36]. 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 3,482  $CFD^p$ s and 2,568  $CIND^p$ s for the DBLP data, with their representatives shown below.

```
 $\phi_1$ : paper (isbn =  $\_$  and booktitle =  $\_$   $\rightarrow$  publisher =  $\_$ )
 $\phi_2$ : paper (title =  $\_$  and year =  $\_$  and booktitle =  $\_$   $\rightarrow$ 
  type =  $\_$ )
 $\phi_3$ : paper (booktitle = 'CleanDB'  $\rightarrow$  year = 2006)
 $\phi_4$ : paper (booktitle = 'VLDB' and yearL =  $\_$   $\rightarrow$  yearR  $\geq$ 
  1975 and yearR  $\leq$  2007)
 $\phi_5$ : paper (booktitle = 'PVLDB' and yearL =  $\_$   $\rightarrow$  yearR  $\geq$ 
  2008)
 $\rho_1$ : paper (crossref, isbn, publisher; type = 'inproceedings'
  and booktitle = 'CIKM-CNIKM')  $\subseteq$  proceeding (key,
  isbn, publisher; year = 2009)
 $\rho_2$ : paper (crossref, isbn, publisher; type = 'inproceedings'
  and booktitle = 'VLDB')  $\subseteq$  proceeding (key, isbn,
  publisher; year  $\geq$  1975 and year  $\leq$  2007)
```



Fig. 6. Efficiency of detecting violations: CFD<sup>p</sup>s versus CFDs.

$\rho_3$ : 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<sup>p</sup>s and CIND<sup>p</sup>s by instantiating the values of their booktitle and year attributes. Observe that  $\phi_1$ - $\phi_3$  and  $\rho_1$  are CFDs and CINDs, respectively. For comparison, we further designed the following CFDs and CINDs to approximate  $\phi_4$ - $\phi_5$  and  $\rho_2$ - $\rho_3$ .

$\phi'_4$ : paper (booktitle = 'VLDB' and year<sub>L</sub> = ' \_ '  $\rightarrow$  year<sub>R</sub> = ' \_ ')

$\phi'_5$ : paper (booktitle = 'PVLDB' and year<sub>L</sub> = ' \_ '  $\rightarrow$  year<sub>R</sub> = ' \_ ')

$\rho'_2$ : paper (crossref, isbn, publisher; type = 'inproceedings' and booktitle = 'VLDB')  $\subseteq$  proceeding (key, isbn, publisher; year = ' \_ ')

$\rho'_3$ : paper (crossref, isbn, publisher; type = 'inproceedings' and booktitle = 'ICDE')  $\subseteq$  proceeding (key, isbn, publisher; year = ' \_ ')

### 6.1.2 Implementation

All the experiments were run within an SQL Server 2012 database installed on a machine with an Intel Core i5 (3.1 GHz) CPU and 8 GB of RAM. Each test was repeated five 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 percent. For easy of comparison, we deliberately dirty the tuples in hcahps of HOSP or paper of DBLP so that using the CFD<sup>p</sup>s and CIND<sup>p</sup>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<sup>p</sup>s and CIND<sup>p</sup>s versus their counterparts CFDs and CINDs, separately and taken together.

**Exp-1.1: CFD<sup>p</sup>s versus CFDs.** (1) To evaluate the impacts of  $|I_1|$ , we fixed *noise%* = 9%, and varied  $|I_1|$  from 10 to 90 K for HOSP (resp. from 100 to 900 K 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

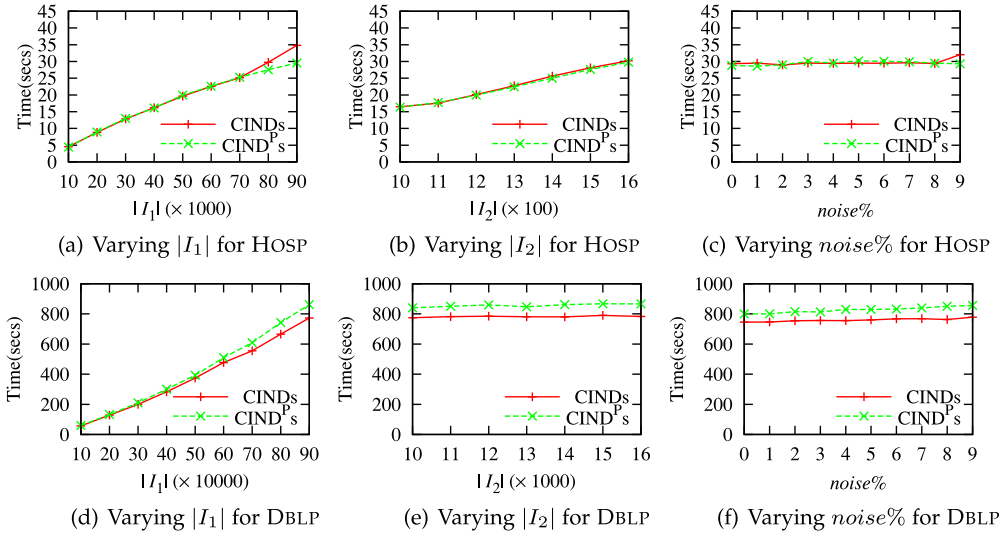
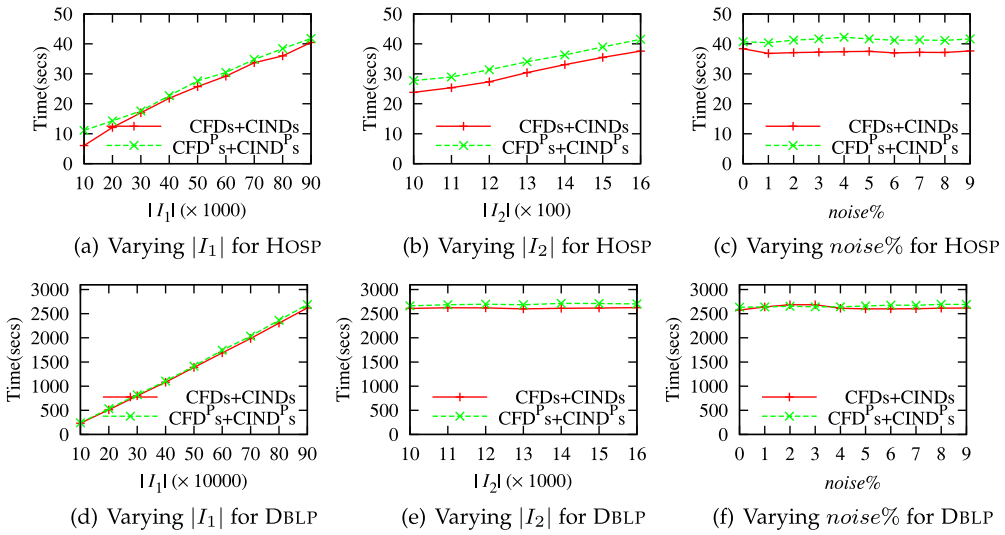
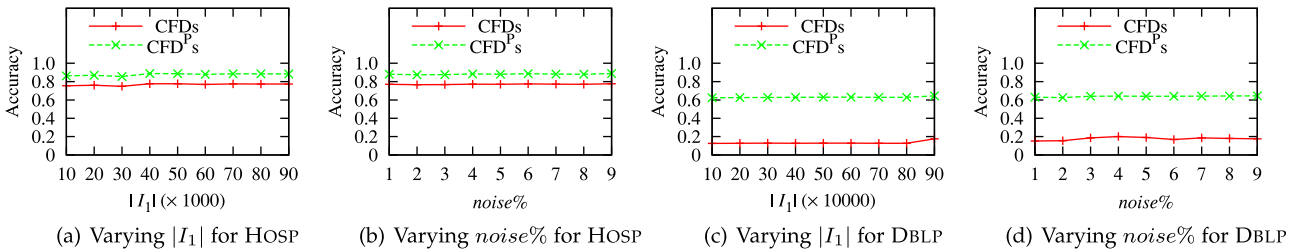
percent. The results are reported in Figs. 6a and 6c and Figs. 6b and 6d, respectively.

The results tell us that for CFDs and CFD<sup>p</sup>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  $\phi'_5$  and  $\phi'_6$  is higher than CFD<sup>p</sup>s  $\phi_5$  and  $\phi_6$ , as they match more  $I_1$  tuples, but the RHS complexity of CFDs  $\phi'_5$  and  $\phi'_6$  is lower than CFD<sup>p</sup>s  $\phi_5$  and  $\phi_6$ , as they are easier to check violations; And (b) the LHS complexity of CFDs  $\phi'_4$  and  $\phi'_5$  is the same as CFD<sup>p</sup>s  $\phi_4$  and  $\phi_5$ , but the RHS complexity of CFDs  $\phi'_4$  and  $\phi'_5$  is similar to CFD<sup>p</sup>s  $\phi_4$  and  $\phi_5$ , as they are easier to check violations. As a combined result, the running time of CFDs is lower than CFD<sup>p</sup>s on HOSP, but close to CFD<sup>p</sup>s on DBLP.

**Exp-1.2: CIND<sup>p</sup>s versus 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 percent. The results are reported in Figs. 7a and 7d, Figs. 7b and 7e, and Figs. 7c and 7f, respectively.

The results tell us that for CINDs and CIND<sup>p</sup>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  $\psi'_3$  is higher than CIND<sup>p</sup>  $\psi_3$ , as they match more  $I_1$  tuples, but the RHS complexity of CIND  $\psi'_3$  is lower than CIND<sup>p</sup>  $\psi_3$ , as they are easier to check violations; And (b) the LHS complexity of CINDs  $\rho'_2$  and  $\rho'_3$  is the same as CIND<sup>p</sup>s  $\rho_2$  and  $\rho_3$ , but the RHS complexity of CINDs  $\rho'_2$  and  $\rho'_3$  is lower than CIND<sup>p</sup>s  $\rho_2$  and  $\rho_3$ , as they are easier to check violations. As a combined result, the running time of CINDs is close to CIND<sup>p</sup>s on HOSP, but is lower on DBLP.

**Exp-1.3: CFD<sup>p</sup>s +CIND<sup>p</sup>s versus CFDs +CINDs.** Using the same setting as Exp-1.2, we evaluated the impacts of  $|I_1|$ ,  $|I_2|$  and *noise%*. The results are reported in Figs. 8a and 8d, Figs. 8b and 8e and Figs. 8c and 8f, respectively. The results show similar findings to Exp-1.1 and Exp-1.2, and are consistent with them.

Fig. 7. Efficiency of detecting violations: CIND<sup>p</sup>'s versus CINDs.Fig. 8. Efficiency of detecting violations: CFD<sup>p</sup>'s + CIND<sup>p</sup>'s versus CFDS + CINDs.Fig. 9. Effectiveness of detecting violations: CFD<sup>p</sup>'s versus CFDS.

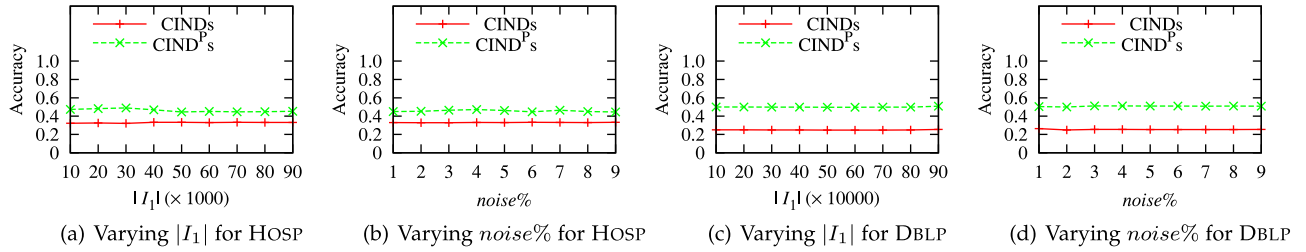
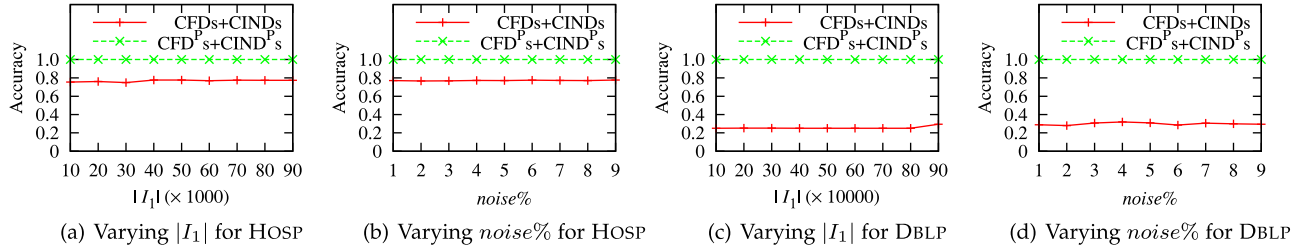
### 6.2.2 Tests of Effectiveness

In the second set of experiments, we evaluated the violation detection effectiveness of CFD<sup>p</sup>'s and CIND<sup>p</sup>'s versus 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<sup>p</sup>'s, CINDs, CIND<sup>p</sup>'s, CFDS + CINDs or CFD<sup>p</sup>'s + CIND<sup>p</sup>'s, denoted by  $x$ , its effectiveness of detecting violations is evaluated with the following measure:

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

**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<sup>p</sup>'s versus CFDS, (b) CIND<sup>p</sup>'s versus CINDs and (c) CFD<sup>p</sup>'s + CIND<sup>p</sup>'s versus CFDS + CINDs, respectively. The results are reported in Figs. 9, 10 and 11, respectively, and are summarized in Table 2.

Fig. 10. Effectiveness of detecting violations: CIND<sup>p</sup>'s versus CINDs.Fig. 11. Effectiveness of detecting violations: CFD<sup>p</sup>'s + CIND<sup>p</sup>'s versus CFDs + CINDs.TABLE 2  
Summary of Violation Detection Accuracy

| Datasets | varying   | CFDs (%)     | CFD <sup>p</sup> 's (%) | CINDs (%)    | CIND <sup>p</sup> 's (%) | CFDs + CINDs (%) | CFD <sup>p</sup> 's + CIND <sup>p</sup> '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  |

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<sup>p</sup>'s, CIND<sup>p</sup>'s and CFD<sup>p</sup>'s + CIND<sup>p</sup>'s obviously outperform their counterparts CFDs, CINDs and CFDs + CINDs, respectively, (3) the increase of effectiveness depends on the increase of the expressive power, and varies from 22 to 75 percent on HOSP and DBLP, and, (4) the increased effectiveness on DBLP is larger than on HOSP, as there are more CFD<sup>p</sup>'s and CIND<sup>p</sup>'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<sup>p</sup>'s and CIND<sup>p</sup>'s is comparable to their CFDs and CINDs counterparts, which is consistent with the static analyses: CFD<sup>p</sup>'s and CIND<sup>p</sup>'s retain the same complexity as their CFDs and CINDs counterparts. (2) CFD<sup>p</sup>'s and CIND<sup>p</sup>'s are able to capture more dirty tuples than CFDs and CINDs, due to the increased expressive power.

## 7 CONCLUSIONS

We have proposed CFD<sup>p</sup>'s and CIND<sup>p</sup>'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<sup>p</sup>'s and CIND<sup>p</sup>'s are more powerful than CFDs and CINDs for detecting errors in real-life data. In addition, the satisfiability and implication problems for CFD<sup>p</sup>'s and CIND<sup>p</sup>'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<sup>p</sup>'s and CIND<sup>p</sup>'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<sup>p</sup>'s, eCFDs [6] and CFD<sup>s</sup> [8]), without increasing the complexity of static analyses. Second, we are to develop effective algorithms for discovering CFD<sup>p</sup>'s and CIND<sup>p</sup>'s, along the same lines as [7], [13], [14]. Third, we plan to extend the methods of [3], [19] to repair data based on CFD<sup>p</sup>'s and CIND<sup>p</sup>'s, instead of using CFDs [3], traditional FDs and INDs [19], denial constraints [21], and aggregate constraints [22].

## ACKNOWLEDGMENTS

S. Ma is supported in part by 973 program 2014CB340300 and NSFC 61322207. W. Fan is supported in part by NSFC 61133002, 973 Program 2012CB316200 and 2014CB340302, ERC-2014-AdG 652976, Guangdong Innovative Research Team Program 2011D005, Shenzhen Peacock Program 1105100030834361, EPSRC EP/J015377/1 and EP/M025268/1, NSF III 1302212, and a Google Faculty Research Award.

## REFERENCES

- [1] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *ACM Trans. Database Syst.*, vol. 33, no. 2, pp. 1–48, 2008.
- [2] S. Ma, W. Fan, and L. Bravo, "Extending inclusion dependencies with conditions," *Theor. Comput. Sci.*, vol. 515, pp. 64–95, 2014.
- [3] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 315–326.



- [4] W. Fan and F. Geerts, *Foundations of Data Quality Management* (series synthesis lectures on data management). San Mateo, CA, USA: Morgan Kaufmann, 2012.
- [5] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Reading, MA, USA: Addison-Wesley, 1995.
- [6] L. Bravo, W. Fan, F. Geerts, and S. Ma, "Increasing the expressivity of conditional functional dependencies without extra complexity," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 516–525.
- [7] L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies," *Proc. VLDB Endowment*, vol. 1, pp. 376–390, 2008.
- [8] W. Chen, W. Fan, and S. Ma, "Incorporating cardinality constraints and synonym rules into conditional functional dependencies," *Inf. Process. Lett.*, vol. 109, no. 14, pp. 783–789, 2009.
- [9] W. Chen, W. Fan, and S. Ma, "Analyses and validation of conditional dependencies with built-in predicates," in *Proc. 20th Int. Conf. Database Expert Syst. Appl.*, 2009, pp. 576–591.
- [10] R. van der Meyden, "The complexity of querying indefinite data about linearly ordered domains," *J. Comput. Syst. Sci.*, vol. 54, no. 1, pp. 113–135, 1997.
- [11] J. Chomicki, "Consistent query answering: Five easy pieces," in *Proc. 11th Int. Conf. Database Theory*, 2007, pp. 1–17.
- [12] F. Chiang and R. J. Miller, "Discovering data quality rules," *Proc. VLDB Endowment*, vol. 1, pp. 1166–1177, 2008.
- [13] A. Zanzi and A. Trombetta, "Discovering non-constant conditional functional dependencies with built-in predicates," in *Proc. 25th Int. Conf. Database Expert Syst. Appl.*, 2014, pp. 35–49.
- [14] J. Bauckmann, Z. Abedjan, U. Leser, H. Müller, and F. Naumann, "Discovering conditional inclusion dependencies," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 2094–2098.
- [15] B. Saha and D. Srivastava, "Data quality: The other face of big data," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 1294–1297.
- [16] M. Arenas, L. E. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," in *Proc. 18th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst.*, 1999, pp. 68–79.
- [17] E. F. Codd, "Relational completeness of data base sublanguages," in *Proc. Data Base Syst.: Courant Comput. Sci. Symposia Ser. 6*, 1972, pp. 65–98.
- [18] M. A. Casanova, R. Fagin, and C. H. Papadimitriou, "Inclusion dependencies and their interaction with functional dependencies," *J. Comput. Syst. Sci.*, vol. 28, no. 1, pp. 29–59, 1984.
- [19] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi, "A cost-based model and effective heuristic for repairing constraints by value modification," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 143–154.
- [20] O. Curi, "Improving the data quality of drug databases using conditional dependencies and ontologies," *J. Data Inf. Quality*, vol. 4, no. 1, p. 3, 2012.
- [21] 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.*, vol. 33, no. 4–5, pp. 407–434, 2008.
- [22] S. Flesca, F. Furfaro, and F. Parisi, "Consistent query answers on numerical databases under aggregate constraints," in *Proc. 10th Int. Conf. Database Program. Lang.*, 2005, pp. 279–294.
- [23] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma, "Dynamic constraints for record matching," *VLDB J.*, vol. 20, no. 4, pp. 495–520, 2011.
- [24] W. Fan, S. Ma, N. Tang, and W. Yu, "Interaction between record matching and data repairing," *J. Data Inf. Quality*, vol. 4, no. 4, p. 16, 2014.
- [25] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, "Towards certain fixes with editing rules and master data," *VLDB J.*, vol. 21, no. 2, pp. 213–238, 2012.
- [26] S. Ginsburg and R. Hull, "Order dependency in the relational model," *Theor. Comput. Sci.*, vol. 26, pp. 149–195, 1983.
- [27] M. Baudinet, J. Chomicki, and P. Wolper, "Constraint-generating dependencies," *J. Comput. Syst. Sci.*, vol. 59, no. 1, pp. 94–115, 1999.
- [28] P. D. Bra and J. Paredaens, "Conditional dependencies for horizontal decompositions," in *Proc. 10th Colloquium Automata, Lang. Programm.*, 1983, pp. 67–82.
- [29] M. J. Maher, "Constrained dependencies," *Theoretical Comput. Sci.*, vol. 173, no. 1, pp. 113–149, 1997.
- [30] M. J. Maher and D. Srivastava, "Chasing constrained tuple-generating dependencies," in *Proc. 15th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.*, 1996, pp. 128–138.
- [31] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery," *ACM Trans. Database Syst.*, vol. 36, no. 3, p. 16, 2011.
- [32] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian, "Metric functional dependencies," in *Proc. IEEE Int. Conf. Data Eng.*, 2009, pp. 1275–1278.
- [33] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.
- [34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.
- [35] [Online]. Available: <https://data.medicare.gov/data/hospital-compare>, 2014.
- [36] [Online]. Available: <http://dblp.uni-trier.de/xml/>, 2014.



**Shuai Ma** received the PhD degrees from the University of Edinburgh in 2010, and Peking University in 2004, respectively. He is a professor in the School of Computer Science and Engineering, Beihang University. He was a postdoctoral research fellow in the database group, University of Edinburgh, a summer intern at Bell labs, Murray Hill, in the summer of 2008, and a visiting researcher at MRSA in 2012. He received 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.



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



**Wenfei Fan** received the PhD degree from the University of Pennsylvania. He is professor (chair) of Web Data Management in the School of Informatics, University of Edinburgh, United Kingdom. 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 and 2015, the Best Paper Award for VLDB 2010, the Roger Needham Award in 2008 (United Kingdom), 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. His current research interests include database theory and systems. He is a fellow of ACM, a fellow of the Royal Society of Edinburgh, United Kingdom, a national professor of the Thousand-Talent Program, and a Yangtze River scholar, China.



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



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

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).