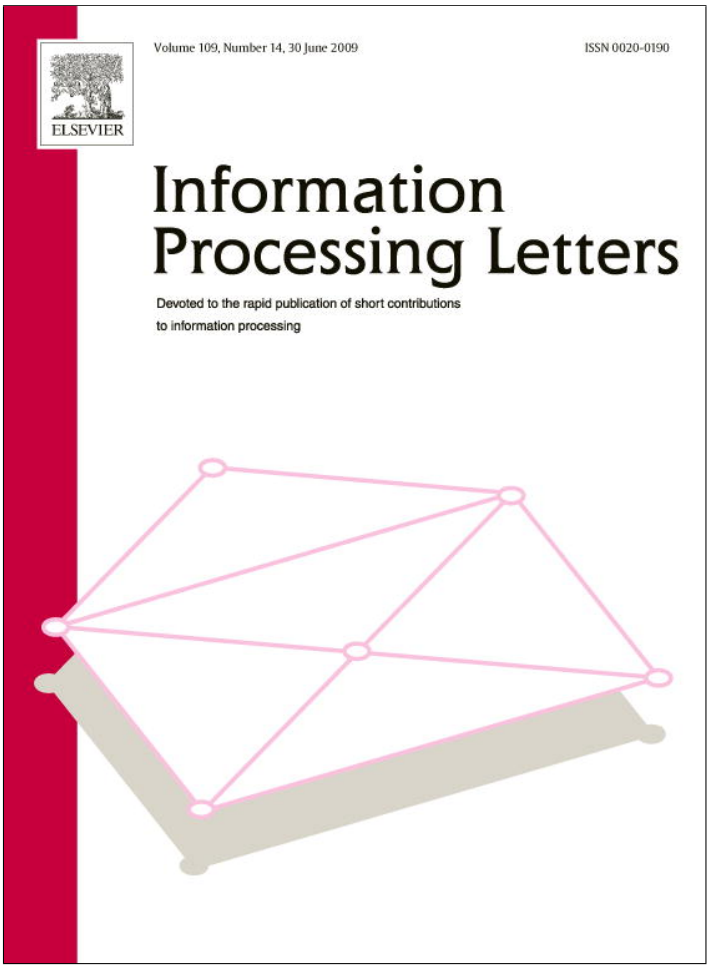
Incorporating cardinality constraints and synonym rules into conditional functional dependencies



Information Processing Letters 109 (2009) 783–789

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Incorporating cardinality constraints and synonym rules into conditional functional dependencies

Wenguang Chen [a], Wenfei Fan [b,c,*], Shuai Ma [b]

[a] *Peking University, China*
[b] *University of Edinburgh, UK*
[c] *Bell Laboratories, USA*

ARTICLE INFO

Article history:
Received 20 December 2008
Received in revised form 22 March 2009
Accepted 23 March 2009
Available online 26 March 2009
Communicated by J. Chomicki

Keywords:
Computational complexity
Databases
Specification languages

ABSTRACT

We propose an extension of conditional functional dependencies (CFDs), denoted by CFD^{c}s, to express cardinality constraints, domain-specific conventions, and patterns of semantically related constants in a uniform constraint formalism. We show that despite the increased expressive power, the satisfiability and implication problems for CFD^{c}s remain NP-complete and coNP-complete, respectively, the same as their counterparts for CFDs. We also identify tractable special cases.

1. Introduction

Conditional functional dependencies (CFDs) have recently been studied for detecting inconsistencies in relational data [14]. These dependencies are an extension of functional dependencies (FDs) by enforcing patterns of semantically related data values. In contrast to traditional FDs that were developed for improving the quality of schema, CFDs aim to improve the quality of the data. That is, CFDs are to be used as data-quality rules such that errors and inconsistencies in the data can be detected as violations of these dependencies.

While CFDs are capable of capturing more errors than traditional FDs, they are not powerful enough to detect certain inconsistencies commonly found in real-life data. To illustrate this, let us consider an example.

Example 1.1. Consider a relation schema:

sale(FN: string, LN: string, street: string, city: string, state: string, country: string, zip: string, item: string, type: string),

where each tuple specifies an item of a certain type purchased by a customer. Each customer is specified by her name (FN, LN) and address (street, city, state, country, zip). An instance D_0 of the sale schema is shown in Fig. 1.

CFDs on sale data include the following:

ϕ_1: $\left([\text{country}, \text{zip}] \to \text{street}, t_p^1\right)$, and $t_p^1 = (\text{UK}, \_ \parallel \_)$
ϕ_2: $\left(\text{country} \to \text{state}, t_p^2\right)$, where $t_p^2 = (\text{UK} \parallel \text{N/A})$.

Here ϕ_1 asserts that for customers in the UK, zip code uniquely determines street. It uses a tuple t_p^1 to specify a pattern: country = UK, zip = '_' and street = '_', where '_' can take an arbitrary value. It is an "FD" that is to hold on the subset of tuples that satisfies the pattern, e.g., $\{t_1, t_3\}$ in D_0, rather than on the entire D_0 (in the US, for example, zip does not determine street). It is not a traditional FD since it is defined with constants. Similarly, ϕ_2 assures that for any address in the UK, state must be N/A (non-applicable); this is enforced by pattern tuple t_p^2: country = UK and state = N/A.

When these CFDs are used as data quality rules, one can see that either t_1 or t_3 is "dirty": they violate the

* Corresponding author.
E-mail address: wenfei@inf.ed.ac.uk (W. Fan).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2009.03.021

	FN	LN	street	city	state	country	zip	item	type
t_1:	Joe	Brady	Mayfield	EDI	N/A	UK	EH4 8LE	CD1	regular
t_2:	Mark	Webber	Crichton	EDI	NY	United Kingdom	EH4 8LE	CD2	sale
t_3:	John	Hull	Queen	EDI	N/A	UK	EH4 8LE	CD3	regular
t_4:	William	Smith	5th Ave	NYC	NY	US	10016	book1	sale
t_5:	Bill	Smith	5th Ave	NYC	NY	US	10016	book2	sale
t_6:	Bill	Smith	5th Ave	NYC	NY	US	10016	book3	sale

Fig. 1. An instance of the sale relation schema.

rule ϕ_1. Indeed, t_1 and t_3 are about customers in the UK and they have the same zip; however, they have different streets.

A closer examination of D_0 reveals that tuple t_2 is not error-free either. Indeed, t_2 is about a transaction for a UK customer, but (a) its state is NY rather than N/A, and (b) while its zip is the same as that of t_1 and t_3, it has a street not found in t_1 or t_3. However, these violations cannot be detected by ϕ_1 and ϕ_2. Indeed, these CFDs are specified with the pattern country = UK, and do not apply to tuples with country = "United Kingdom". Although UK and United Kingdom refer to the same country, they are not treated as equal by the equality operator adopted by CFDs and FDs. In other words, CFDs and FDs do not observe domain-specific abbreviations and conventions.

Another issue concerns *cardinality constraints* commonly found in practice, which require that the number of tuples with a certain pattern does not exceed a predefined bound. An example is that each customer is allowed to purchase at most two distinct items on sale (with type = sale). As another example, on a school database, one may want to specify that a CS student can register for at most six courses each semester. These constraints can be expressed as neither FDs nor CFDs.

These practical concerns highlight the following questions. Can one extend CFDs to express cardinality constraints and synonym rules (domains-specific abbreviations and conventions)? Can we find an extension such that it does not increase the complexity for reasoning about these dependencies? Indeed, we want a balance between the expressive power needed to deal with these issues, and the complexity for static analyses of the dependencies.

Contributions. We answer these questions in this paper, by providing the following.

(1) We propose an extension of CFDs, denoted by CFD^{c}s, that is able to express cardinality constraints, synonym rules and patterns of semantically related values of CFDs in a uniform constraint formalism. For example, all constraints we have seen so far can be expressed as CFD^{c}s.
(2) We establish complexity bounds for the satisfiability problem and the implication problem associated with CFD^{c}s. The satisfiability problem is to determine whether a set Σ of CFD^{c}s has a nonempty model, i.e., whether the data quality rules in Σ make sense. The implication problem is to decide whether a set Σ of CFD^{c}s entails another CFD^c φ, i.e., whether the rule φ is redundant given the rules in Σ. We show that despite the increased expressive power of CFD^{c}s, their satisfiability and implication problems are NP-complete and coNP-complete, respectively, the same as their counterparts for CFDs [14].
(3) We identify special cases where the satisfiability and implication analyses of CFD^{c}s are in PTIME. That is, in these practical settings we are able to reason about CFD^{c}s efficiently.

We contend that CFD^{c}s yield a better tool than CFDs for detecting errors, without increasing the complexity of static analyses.

Related work. To our knowledge, no previous work has studied extensions of CFDs to capture cardinality constraints and synonym rules.

Constraint-based data cleaning was introduced in [4], which proposed to use dependencies, e.g., FDs, inclusion dependencies (INDs) and denial constraints, to detect errors in real-life data (see, e.g., [12] for a comprehensive survey). As an extension of traditional FDs, CFDs were developed in [14], which showed that the satisfiability problem and implication problem for CFDs are NP-complete and coNP-complete, respectively. There have been extensions of CFDs to support disjunction and negation [9], and ranges of values in pattern tuples [16]. These extensions address issues quite different from the focus of CFD^{c}s, and will be further discussed in Section 5. Algorithms have been developed for discovering CFDs [11,16] and for repairing data based on CFDs [13]. There have also been a variety of extensions of FDs [6,8,19] (see [14] for a detailed discussion about the differences between these extensions and CFDs). To the best of our knowledge, no previous work has studied how to extend CFDs or FDs to express cardinality constraints, abbreviations and conventions.

Synonym rules have been studied for record matching [2,3] in the form of transformation rules. However, no previous work has studied how to express these in dependencies, or their impact on the static analyses of dependencies.

Cardinality constraints have been studied for relational data [18] to constrain the domains of attributes, and for object-oriented databases to restrict the extents of classes [10]. Numerical dependencies [17], which generalize FDs with cardinality constraints, have also been proposed for schema design. These constraints differ from CFD^{c}s in that they cannot constrain tuples with a pattern specified in terms of constants. Query answering has been investigated for aggregate queries, FDs and denial constraints [5,7], which differ from this work in that neither these dependencies can express cardinality constraints, nor

the impact of cardinality constraints on the satisfiability and implication analyses has been considered.

Organization. Section 2 defines CFD^{c}s, followed by their satisfiability and implication analyses in Sections 3 and 4, respectively. Open issues are discussed in Section 5.

2. CFD^{c}s: An extension of CFDs

Consider a relation schema R defined over a set of attributes, denoted by attr(R). For each attribute $A \in \text{attr}(R)$, its domain is specified in R, denoted as $dom(A)$. As will be seen in Sections 3 and 4, the domains of attributes have substantial impact on the complexity of satisfiability and implication analyses of CFD^{c}s.

CFD^{c}s. A CFD^c φ defined on schema R is a triple $R(X \to Y, t_p, c)$, where (1) $X \to Y$ is a standard FD, referred to as the FD *embedded in* φ; (2) t_p is a tuple with attributes in X and Y, referred to as the *pattern tuple* of φ, where for each A in $X \cup Y$, $t_p[A]$ is either a constant 'a' in $dom(A)$, or an unnamed (yet marked) variable '_' that draws values from $dom(A)$; and (3) c is a positive integer. We refer to φ also as a *conditional functional dependency*.

Intuitively, t_p specifies a pattern of semantically related values for X and Y attributes: for any tuple t in an instance of R, if $t[X]$ has the pattern $t_p[X]$, then $t[Y]$ must observe the pattern $t_p[Y]$. Furthermore, for all those tuples t such that $t[X]$ has pattern $t_p[X]$, if we group $t[Y]$ values by $t[X]$, then the number of distinct values in (i.e., the cardinality of) each group is not allowed to exceed the bound c. In particular, when $c = 1$, $t[X]$ uniquely determines $t[Y]$, i.e., the FD embedded in φ is enforced on those tuples having a $t_p[X]$ pattern.

If A occurs in both X and Y, we use $t_p[A_L]$ and $t_p[A_R]$ to indicate its occurrence in X and Y, respectively. We separate the X and Y attributes in t_p with '$\|$', and denote X as LHS(φ) and Y as RHS(φ). We write φ as $(X \to Y, t_p, c)$ when R is clear from the context.

Example 2.1. CFDs ϕ_1 and ϕ_2 of Example 1.1 can be expressed as CFD^{c}s below, in which t_p^1 and t_p^2 are pattern tuples given in Example 1.1:

$\varphi_1 := \big([\text{country}, \text{zip}] \to \text{street}, t_p^1, 1\big)$,
φ_2: $\big(\text{country} \to \text{state}, t_p^2, 1\big)$.

The cardinality constraint described in Example 1.1 can also be written as a CFD^c φ_3: $(\text{fd}, t_p^3, 2)$, where FD fd and pattern tuple t_p^3 are:

$\text{fd} : \text{FN}, \text{LN}, \text{street}, \text{city}, \text{state}, \text{country}, \text{zip}, \text{type} \to \text{item}$,
$t_p^3 = (\_, \_, \_, \_, \_, \_, \_, \text{sale} \,\|\, \_)$,

assuring that no customer may purchase more than two distinct items with type = sale.

Semantics of CFD^{c}s. To give the semantics of CFD^{c}s, we first extend the equality relation and revise the match operator of [14].

An extension of equality. We use a finite binary relation R_c to capture synonym rules. For values a and b, $R_c(a, b)$ indicates that a and b refer to the same real-world entity. For example, R_c("William", "Bill") and R_c("United Kingdom", "UK"). We assume without loss of generality that R_c is symmetric: if $R_c(a, b)$ then $R_c(b, a)$. However, R_c may *not* be transitive: from R_c("New York State", "NY") and R_c("NY", "New York City") it does not follow that R_c("New York State", "New York City").

In the sequel we assume that R_c is predefined, as commonly found in practice.

We define a binary operator $\doteq$ on constants such that for any values a and b, $a \doteq b$ iff (1) $R_c(a, b)$ or $a = b$, (2) $b \doteq a$, or (3) there exists a value c such that $a \doteq c$ and $b = c$. For example, "United Kingdom" $\doteq$ "UK".

The operator $\doteq$ naturally extends to tuples: $(a_1, \ldots, a_k) \doteq (b_1, \ldots, b_k)$ iff for all $i \in [1, k]$, $a_i \doteq b_i$. Observe that given a fixed R_c, whether $a \doteq b$ can be decided in polynomial time.

Matching operator. We revise the binary operator $\asymp$ of [14] defined on constants and '_' as follows: $\eta_1 \asymp \eta_2$ if either (a) η_1 and η_2 are constants and $\eta_1 \doteq \eta_2$, or (b) one of η_1, η_2 is '_'. The operator $\asymp$ extends to tuples, e.g., $(a, b) \asymp (\_, b)$ but $(a, b) \not\asymp (\_, c)$ if $b \not\doteq c$.

Semantics. Based on $\doteq$ and $\asymp$, we now give the semantics of CFD^c $\varphi = R(X \to Y, t_p, c)$.

An instance D of schema R *satisfies* φ, denoted by $D \models \varphi$, iff for each tuple t in D, if $t[X] \asymp t_p[X]$, then (1) $t[Y] \asymp t_p[Y]$, and (2) $|\pi_Y(\sigma_{X \doteq t[X]} D)| \leqslant c$, i.e., for all tuples t' in D such that $t'[X] \doteq t[X]$, there exist at most c distinct $t'[Y]$ values. Here π and σ are the projection and selection operators in relational algebra, respectively; and $|S|$ denotes the cardinality of a set S in which no two elements a, b are comparable by $a \doteq b$.

Intuitively, φ is a constraint defined on the set of tuples $D_\varphi = \{t \mid t \in D,\ t[X] \asymp t_p[X]\}$ such that (a) for each $t \in D_\varphi$, the pattern $t_p[Y]$ is enforced on $t[Y]$; (b) for each set of tuples in D_φ grouped by X attribute values, the number of their distinct Y values is bounded by the constant c; that is, φ expresses a *cardinality constraint* on the Y values of those tuples grouped by X; and (c) synonym rules are captured by the extension $\doteq$ of the equality relation. Note that φ is defined on the subset D_φ of D identified by $t_p[X]$, rather than on the entire D.

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

Example 2.2. Assume that R_c consists of ("United Kingdom", "UK") and ("William", "Bill"). Recall instance D_0 of Fig. 1 and CFD^{c}s φ_1, φ_2 and φ_3 of Example 2.1. Observe the following: (a) tuple t_2 in D_0 violates φ_2, since $t_2[\text{country}] \asymp \text{UK}$ but $t_2[\text{state}] \not\asymp \text{N/A}$; (b) t_1, t_2 and t_3 violate φ_1 since they are UK records with the same zip code, but they have different streets; (c) t_4, t_5 and t_6 violate φ_3, since they agree on name and address (note that William $\doteq$ Bill), all have type = sale, but they have three distinct items, beyond the bound 2.

Three special cases of CFD^{c}s are worth mentioning. (a) Traditional FDs are CFD^{c}s in which c is 1 and the pattern tuple consists of '_' only. (b) CFDs of [14] are CFD^{c}s in which c is fixed to be 1. (c) *Constant* CFD^{c}s are CFD^{c}s in which the pattern tuples consist of constants only, i.e., they do not contain '_'.

3. The satisfiability analysis

A central technical problem associated with CFD^{c}s is the satisfiability problem.

The *satisfiability problem* for CFD^{c}s is to determine, given a set Σ of CFD^{c}s on a schema R, whether or not there exists a nonempty instance D of R such that $D \models \Sigma$. The set Σ is said to be *satisfiable* if such an instance exists.

Intuitively, the satisfiability problem is to decide whether a set of CFD^{c}s makes sense or not. When CFD^{c}s are used as data quality rules, the satisfiability analysis helps us detect whether the rules are dirty themselves.

Any set of FDs is satisfied by a nonempty relation. In contrast, the satisfiability problem becomes NP-complete for CFDs [14]. Since CFD^{c}s subsume CFDs, the satisfiability problem for CFD^{c}s is at least as hard as for CFDs.

Example 3.1. Consider a schema $R(A, B, C)$, and a set Σ_1 consisting of three CFD^{c}s defined on R: $\psi_1 = (A \to B, (\mathsf{true} \parallel b), 1)$, $\psi_2 = (A \to B, (\mathsf{false} \parallel b), 1)$, and $\psi_3 = (C \to B, (\_ \parallel b'), 1)$, where $dom(A)$ is Boolean, and $b \neq b'$. Then Σ_1 is not satisfiable. Indeed, for any nonempty instance D of R and any tuple t in D, ψ_3 requires $t[B]$ to be b' no matter what value $t[C]$ is, whereas ψ_1 and ψ_2 force $t[B]$ to be b no matter whether $t[A]$ is true or false.

The intractability. Despite the increased expressive power, CFD^{c}s do not complicate the satisfiability analysis. Indeed, the satisfiability problem for CFD^{c}s remains in NP. The proof for the result below is an extension of Theorem 3.2 in [14], its counterpart for CFDs.

Theorem 3.1. *The satisfiability problem for* CFD^c*s is* NP*-complete.*

Proof. It is known that the satisfiability problem is already NP-hard even for constant CFDs [14]. Since CFD^{c}s subsume CFDs, the NP lower bound for CFDs carries over to CFD^{c}s.

We show the upper bound by presenting an NP algorithm that, given a set Σ of CFD^{c}s on a schema R, checks whether Σ is satisfiable. Similar to CFDs [14], CFD^{c}s have a *small model property*: if there is a nonempty instance D of R such that $D \models \Sigma$, then for any $t \in D$, $\{t\}$ is an instance of R and $\{t\} \models \Sigma$. Thus it suffices to consider single-tuple instances $\{t\}$ for deciding whether Σ is satisfiable.

Assume without loss of generality that $\mathsf{attr}(R) = \{A_1, \ldots, A_n\}$. For each $i \in [1, n]$, define the active domain of A_i to be a set $adom(A_i)$ consisting of all constants of $t_p[A_i]$ for all pattern tuples t_p in Σ, plus an extra distinct value in $dom(A_i)$ (if there exists one). Then it is easy to verify that Σ is satisfiable iff there exists a mapping ρ that assigns a value in $adom(A_i)$ to $t[A_i]$ for each $i \in [1, n]$ such that $D = \{(\rho(t[A_1]), \ldots, \rho(t[A_n]))\}$ and $D \models \Sigma$.

Based on these, we give the NP algorithm as follows: (a) Guess a single tuple t of R such that $t[A_i] \in adom(A_i)$ for each $i \in [1, n]$. (b) Check whether $\{t\} \models \Sigma$. If so it returns "yes", and otherwise it repeats steps (a) and (b). Note that step (b) involves checking whether $x \doteq y$, which can be done in PTIME in the sizes of Σ and R_c, where R_c is the relation given in the definition of $\doteq$. Hence the algorithm is in NP, and so is the satisfiability problem. □

A tractable case. As shown by Example 3.1, the complexity is introduced by attributes in CFD^{c}s with a finite domain. This motivates us to consider the following special case.

A set Σ of CFD^c is said to be *bounded by a constant k* if at most k attributes in the CFD^{c}s of Σ have a finite domain. In particular, when $k = 0$, all CFD^{c}s in Σ are defined in terms of attributes with an infinite domain.

Bounded CFD^{c}s make our lives much easier. Indeed, an extension of the proof of Proposition 3.5 in [14] suffices to show the following.

Proposition 3.2. *It is in PTIME to determine whether a set Σ of* CFD^c*s is satisfiable if Σ is bounded by a constant k.*

Proof. When Σ is bounded by k, we develop a PTIME algorithm to determine whether Σ is satisfiable, which is based on a modified chase (see, e.g., [1] for the chase), and the small model property identified in the proof of Theorem 3.1. The algorithm is an extension of the one for CFDs (Proposition 3.5 in [14]) to further deal with finite domain attributes and the $\doteq$ operator. Assume without loss of generality that Σ is defined on a schema R, and only attributes A_i in CFD^{c}s of Σ have a finite domain, for $i \in [1, k]$.

The algorithm checks whether there exists a tuple t of R such that $t \models \Sigma$. Initially $t[A]$ is a distinct variable x_A for each $A \in \mathsf{attr}(R)$. For all $i \in [1, k]$ and for each value in $dom(A_i)$ assigned to x_{A_i}, the algorithm does the following.

(a) For each CFD^c $\phi = R(X \to Y, t_p, c)$ in Σ, chase t using ϕ: if $t[X] \asymp t_p[X]$, then change $t[Y]$ such that $t[Y] \asymp t_p[Y]$ as long as $t[Y]$ does not already contain a constant that does not match the corresponding field in $t_p[Y]$.

Here we extend the match operator $\asymp$ to accommodate variables x_B: $x_B \asymp \_$, but $x_B \not\asymp \eta$ when η is a constant or a variable.

(b) For each attribute $B \in \mathsf{attr}(R)$, if $t[B]$ is still x_B after step (a), assign a distinct value from $dom(B)$ to x_B, which does not appear in Σ and R_c; note that $dom(B)$ must be infinite in this case by the definition of t.

(c) If $t \models \Sigma$ then return "yes"; "no" is returned if for all possible valuations to x_{A_i} for $i \in [1, k]$, it cannot instantiate t such that $t \models \Sigma$.

The algorithm is in $O(|\Sigma|^2 |R_c| m^k)$ time, i.e., in PTIME when k is fixed, where $|\Sigma|$ is the size of Σ, $|R_c|$ is the size of R_c (in the definition of $\doteq$), and m is the maximum cardinality of finite domains $adom(A_i)$ for $i \in [1, k]$.

We next show that the algorithm returns "yes" if and only if Σ is satisfiable.

If the algorithm returns "yes", there exists a tuple t such that $t \models \Sigma$. Thus Σ is satisfiable.

Conversely, if Σ is satisfiable, there exists a tuple t such $t \models \Sigma$. We show that the algorithm returns "yes".

Initialize a tuple t' such that $t'[A_i] = t[A_i]$ for $i \in [1,k]$, and $t'[A] = x_A$ for the rest of attributes $A \in \text{attr}(R)$. After step (a), for each attribute $A \in \text{attr}(R)$, if $t'[A]$ is a constant, then $t'[A] \doteq t[A]$. Moreover, there exist no conflicts since $t \models \Sigma$. The assignments at step (b) are irrelevant since $t'[B]$'s instantiated at that step are not constrained by pattern tuples in Σ, and thus have no impact on whether $\{t'\}$ satisfies Σ. Thus after step (b), $\{t'\} \models \Sigma$, and the algorithm returns "yes". □

4. The implication analysis

We next investigate another central technical problem associated with CFD^{c}s.

Consider a set Σ of CFD^{c}s and a single CFD^c defined on the same schema R. We say that Σ *implies* φ, denoted by $\Sigma \models \varphi$, iff for all instances D of R, if $D \models \Sigma$ then $D \models \varphi$. We consider without loss of generality satisfiable Σ only.

The *implication problem* for CFD^{c}s is to determine, given a set Σ of CFD^{c}s and a CFD^c defined on the same schema, whether $\Sigma \models \varphi$.

The implication analysis helps us identify and eliminate redundant data quality rules.

As examples of the implication analysis, we present two simple results.

Proposition 4.1. *For any* CFD^c*s of the form*:

φ: $R(X \to Y, t_p, c)$, $\quad \varphi'$: $R(X \to Y, t_p, c')$

(a) $\varphi \models \varphi'$ *if* $c \leqslant c'$; *and*
(b) *if* φ *is a constant* CFD^c, $\varphi \models \varphi'$ *even when* $c' = 1$ *and* $c > c'$.

Proof. (a) This can be easily verified by the definition of CFD^{c}s. (b) We show that for any instance D of R, if $D \models \varphi$ then $D \models \varphi'$. Observe that for any tuple $t \in D$, if $t[X] \doteq t_p[X]$, then $t[Y] \doteq t_p[Y]$. Hence for all tuples t' in D, if $t'[X] \doteq t[X]$, then $t'[Y] \doteq t_p[Y]$, i.e., $|\pi_Y(\sigma_{X \doteq t[X]} D)| \leqslant 1$. Thus $D \models \varphi'$. □

The intractability. We know that the implication problem for CFDs is coNP-complete [14]. Below we show that the upper bound remains intact for CFD^{c}s, along the same lines as its CFD counterpart (Theorem 4.3 in [14]).

In the rest of the section we consider a set Σ of CFD^{c}s and a CFD^c $\varphi = R(X \to Y, t_p, c)$ such that c is bounded by a polynomial in the sizes of Σ and φ. This assumption is acceptable since in practice, c is typically fairly small.

Theorem 4.2. *The implication problem for* CFD^c*s is* coNP*-complete.*

Proof. The implication problem for constant CFDs is coNP-hard [14]. The lower bound carries over to CFD^{c}s, which subsume CFDs.

We show that the problem is in coNP by presenting an NP algorithm for its complement, i.e., for deciding whether $\Sigma \not\models \varphi$. The algorithm is based on a small model property: if $\varphi = R(X \to Y, t_p, c)$ and $\Sigma \not\models \varphi$, then there exists an instance D of R with at most $c+1$ tuples such that $D \models \Sigma$ and $D \not\models \varphi$. That is, D consists of $c+1$ tuples $t_1, \ldots, t_{c+1}$ such that for all $i, j \in [1, c+1]$, $t_i[X] \asymp t_p[X]$ and $t_i[X] \doteq t_j[X]$, but either there exists $l \in [1, c+1]$ such that $t_l[Y] \not\asymp t_p[Y]$, or for all $i \neq j$, $t_i[Y] \not\doteq t_j[Y]$. Thus it suffices to consider instances D with $c+1$ tuples for deciding whether $\Sigma \not\models \varphi$.

Assume that $\text{attr}(R) = \{A_1, \ldots, A_n\}$. For each $i \in [1,n]$, let $adom(A_i)$ be a set consisting of (a) all constants of $t_p[A_i]$ for all pattern tuples t_p in $\Sigma \cup \{\varphi\}$, and (b) $c+1$ extra distinct values in $dom(A_i)$ if they exist; if $dom(A_i)$ is finite and does not have $c+1$ extra values, let $adom(A_i)$ be $dom(A_i)$. Then one can verify that $\Sigma \not\models \varphi$ iff there exist mappings $\rho_1, \ldots, \rho_{c+1}$ such that ρ_i maps $t[A_j]$ to a value in $adom(A_j)$ for each $j \in [1,n]$, $D = \{(\rho_1(t[A_1]), \ldots, \rho_1(t[A_n])), \ldots, (\rho_{c+1}(t[A_1]), \ldots, \rho_{c+1}(t[A_n]))\}$, $D \models \Sigma$ and $D \not\models \varphi$.

Based on these, we give the NP algorithm as follows: (a) Guess $c+1$ tuples $t_1, \ldots, t_{c+1}$ of R such that $t_j[A_i] \in adom(A_i)$ for each $i \in [1,n]$ and $j \in [1, c+1]$. (b) Check whether $\{t_1, \ldots, t_{c+1}\}$ satisfies Σ, but not φ. If so the algorithm returns "yes", and otherwise it repeats steps (a) and (b). As argued in the proof of Theorem 3.1, step (b) can be done in PTIME in the sizes of Σ, φ and R_c. Furthermore, c is bounded by a polynomial by assumption. As a result, the algorithm is in NP and thus the implication problem is in coNP. □

Special cases. Proposition 3.2 shows that for a set of CFD^{c}s bounded by a constant k, the satisfiability analysis is in PTIME. This is no longer the case for the implication problem.

Theorem 4.3. *It is* coNP*-complete to decide, given* CFD^c*s* Σ *and* φ, *whether* $\Sigma \models \varphi$ *when* $\Sigma \cup \{\varphi\}$ *is bounded by a constant* $k = 3$.

Proof. The problem is in coNP by Theorem 4.2. We show that it is coNP-hard by reduction from 3SAT to the complement of the problem (i.e., to decide whether $\Sigma \not\models \varphi$), where 3SAT is NP-complete (cf. [15]). Consider an instance $\phi = C_1 \wedge \cdots \wedge C_n$ of 3SAT, where all the variables in ϕ are $x_1, \ldots, x_m$, C_j is of the form $y_{j_1} \vee y_{j_2} \vee y_{j_3}$, and moreover, for $i \in [1,3]$, y_{j_i} is either $x_{p_{ji}}$ or $\overline{x_{p_{ji}}}$ for $p_{ji} \in [1,m]$; here we use $x_{p_{ji}}$ to indicate the occurrence of a variable in literal i of clause C_j. Given ϕ, we construct a relation schema R, an empty relation R_c, and a set $\Sigma \cup \{\varphi\}$ of CFD^{c}s defined on R, such that ϕ is satisfiable iff $\Sigma \not\models \varphi$.

(1) We define schema $R(C, V_c, X, V_x, Z)$, where $dom(C) = \{1, \ldots, n\}$, $dom(V_c) = \{\langle b_1 b_2 b_3 \rangle \mid b_1, b_2, b_3 \in \{0,1\}\}$, $dom(X) = \{x_1, \ldots, x_m\}$, which is the set of variables in ϕ, and moreover, both $dom(V_x)$ and $dom(Z)$ are integer. Intuitively, for each R tuple t, $t[C]$, $t[V_c]$, $t[X]$, $t[V_x]$ and $t[Z]$ specify a clause C, a truth assignment ξ (one of the eight to its three variables), one of the three variables in C, the truth value of the variable and the truth value of C determined by ξ.

(2) Let the set Σ of CFD^{c}s be $\Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$.

(a) Σ_1 encodes the relationships among attributes C, V_c, X and V_x. For each variable in a clause C_j ($1 \leqslant j \leqslant n$) and each value $\langle b_1 b_2 b_3 \rangle$ in $dom(V_c)$, there is a CFD^c in Σ_1.

Thus there are $3*8$ CFD^{c}s for each clause C_j in Σ_1, and in total, there are $24*n$ CFD^{c}s in Σ_1.

Each CFD^c for clause $C_j = y_{j_1} \vee y_{j_2} \vee y_{j_3}$ is of the form of $R((C, V_c, X \to V_x), t_p, 1)$ such that $t_p[C] = j$, $t_p[V_c] = \langle b_1 b_2 b_3 \rangle$, and $t_p[X] = x_{p_{ji}}$ $(1 \leqslant i \leqslant 3)$. The value of $t_p[V_x]$ is decided by the value of $t_p[V_c]$ such that $t_p[V_x] = b_i$ if $y_{j_i} = x_{p_{ji}}$ and otherwise $t_p[V_x] = 1 - b_i$ if $y_{j_i} = \overline{x_{p_{ji}}}$.

For example, if $C_j = x_{p_{j1}} \vee \overline{x_{p_{j2}}} \vee x_{p_{j3}}$ such that $1 \leqslant p_{j1}, p_{j2}, p_{j3} \leqslant m$, then some possible pattern tuples are $(j, \langle 010 \rangle, x_{p_{j1}}, 0)$, $(j, \langle 010 \rangle, x_{p_{j2}}, 0)$, and $(j, \langle 010 \rangle, x_{p_{j3}}, 0)$.

(b) Σ_2 prevents certain variables from appearing in clauses. For each clause C_j and each variable x_i *not* in C_j, two CFD^{c}s are included in Σ_2: $\mu_{j,i,1} = R((C, X \to Z), (j, x_i \parallel 1), 1)$ and $\mu_{j,i,2} = R((C, X \to Z), (j, x_i \parallel 0), 1)$. Thus no tuple t satisfies $t[C] = j$ and $t[X] = x_i$, since otherwise $\mu_{j,i,1}$ forces $t[Z] = 1$ and $\mu_{j,i,2}$ forces $t[Z] = 0$. There are $(m-3)*n$ CFD^{c}s in Σ_2.

(c) Σ_3 encodes the relationship between the truth assignment V_c of clause C and its corresponding truth value Z of C. For clause C_j and each $h \in dom(V_c)$, $\omega_h = R(V_c \to Z, t_{p_h}, 1)$ is in Σ_3, where $t_{p_h}[V_c] = h$, $t_{p_h}[Z] = 0$ if $h = \langle 000 \rangle$, i.e., C is not satisfied by the corresponding truth assignment h, and $t_{p_h}[Z] = 1$ otherwise. In total, Σ_3 consists of eight CFD^{c}s.

(d) Σ_4 includes $\mu_1 = R(C \to V_c, (\_ \parallel \_), 1)$ and $\mu_2 = R(X \to V_x, (\_ \parallel \_), 1)$, ensuring that for each clause C and each variable X, there is at most one truth assignment.

(3) CFD^c φ is defined as $R((Z \to C, X), (1 \parallel \_, \_), 3*n-1)$. Intuitively, φ assures that no more than $3*n-1$ tuples in an instance of R can have truth value 1 for their clauses.

Observe that Σ consists of $(m+21)*n+10$ CFD^{c}s. Thus the reduction is in PTIME.

We now show that ϕ is satisfiable iff $\Sigma \not\models \varphi$. Suppose first that ϕ is satisfiable. Then there exists a truth assignment ρ that makes ϕ true. Based on ρ, we construct an instance D of R with $3*n$ tuples as follows. For each clause $C_j = y_{j_1} \vee y_{j_2} \vee y_{j_3}$ and each variable $x_{p_{ji}}$ $(i \in [1,3])$ in C_j, we create a tuple t, where (a) $t[C] = j$; (b) $t[X] = x_{p_{ji}}$; (c) $t[Z] = 1$; (d) $t[V_x] = 1$ if $x_{p_{ji}}$ is assigned true by ρ, and otherwise $t[V_x] = 0$; (e) $t[V_c] = \langle b_1 b_2 b_2 \rangle$ such that for each $i \in [1,3]$, $b_i = 1$ if y_{j_i} is assigned true by ρ, and otherwise $b_i = 0$. That is, $t[V_c]$ is determined by ρ to all of its three variables. Observe that $D \models \Sigma$ but $D \not\models \varphi$. Hence $\Sigma \not\models \varphi$.

Conversely, if $\Sigma \not\models \varphi$, then there exists an instance D of R consisting of $3*n$ tuples such that $D \models \Sigma$ but $D \not\models \varphi$. Observe that there exist at most n distinct values for attribute C, and each value of C can be associated with at most three distinct values of attribute X. Based on this, we define a truth assignment ρ such that $\rho(x_i) = \text{true}$ if $\pi_{V_x}(\sigma_{X=x_i} D) = \{1\}$ and $\rho(x_i) = \text{false}$ otherwise. Observe that by $D \models \Sigma$, (a) $\pi_{V_x}(\sigma_{X=x_i} D)$ $(i \in [1,m])$ contains exactly one element, (b) $\pi_{V_c}(\sigma_{C=j} D)$ $(j \in [1,n])$ contains one element, and (c) $\pi_{CV_cV_x}(D)$ has $3*n$ elements. Indeed, since $D \models \Sigma$, the truth assignment ρ makes ϕ true. Thus ϕ is satisfiable. □

The proof of Theorem 4.3 actually yields a stronger result. Recall that a CFD^c $R(X \to Y, t_p, c)$ is a CFD of [14] when $c = 1$.

Corollary 4.4. *It remains* coNP*-complete to decide, given a set Σ of* CFDs *and a* CFD^c φ*, whether $\Sigma \models \varphi$ when $\Sigma \cup \{\varphi\}$ is bounded by a constant $k = 3$.*

Not all is lost. Below we identify two tractable special cases. It should be remarked that while the second case below can find a counterpart for CFDs (Corollary 4.4 of [14]), its proof is quite different from that of [14]. Putting this and Corollary 4.4 together, one can tell that the extension of the equality operator and the presence of cardinality constraints take their toll in the implication analysis.

Proposition 4.5. *It is in PTIME to decide, given a set Σ of* CFD^c*s and a* CFD^c φ*, whether $\Sigma \models \varphi$ when $\Sigma \cup \{\varphi\}$ is bounded by a constant k and one of the following conditions holds*:

1. *φ is a* CFD *while Σ is a set of* CFD^c*s*; *or*
2. *Σ is a set of* CFDs*, φ is a* CFD^c *and $k = 0$, i.e., all attributes in Σ or φ have an infinite domain.*

Proof. Observe that $\Sigma \not\models \varphi$ iff there exists a nonempty instance D of the schema R on which Σ and φ are defined, such that $D \models \Sigma \cup \{\neg\varphi\}$. Thus it suffices to develop a PTIME algorithm to check the satisfiability of $\Sigma \cup \{\neg\varphi\}$.

Assume that φ is $R(X \to Y, t_p, c)$.

(1) Since φ is a CFD, the proof of Theorem 4.2 tells us that $\Sigma \cup \{\neg\varphi\}$ is satisfiable iff there exists an instance D_1 of R such that D_1 consists of two tuples t_1 and t_2, $D_1 \models \Sigma$, $t_1[X] \asymp t_p[X]$ and $t_1[X] \doteq t_2[X]$, but either $t_1[Y] \neq t_2[Y]$, or there exists $l \in [1,2]$ such that $t_l[Y] \not\asymp t_p[Y]$. In light of these, a minor extension of the PTIME algorithm given in the proof of Proposition 3.2 suffices to check whether $\Sigma \cup \{\neg\varphi\}$ is satisfiable. Assume without loss of generality that Σ is defined on a schema R, and only attributes A_i in CFD^{c}s of Σ have a finite domain, for $i \in [1,k]$.

The algorithm checks whether there exists an instance $D_1 = \{t_1, t_2\}$ such that $D_1 \models \Sigma$, but $D_1 \not\models \varphi$. Initially, for each attribute $A \in X$, $t_1[A]$ and $t_2[A]$ are the same distinct variable x_A if $t_p[A]$ is '_', and $t_1[A] = t_2[A] = t_p[A]$ if $t_p[A]$ is a constant. For each other attribute A in attr(R) (but not in X), $t_1[A]$ and $t_2[A]$ are two distinct variables x_A and y_A, respectively.

For all $i \in [1,k]$ and for each instantiation of variables x_{A_i} and y_{A_i} with values in $dom(A_i)$, the algorithm does the following.

(a) For each CFD^c $\varphi' = R(X' \to Y', t'_p, c')$ in Σ, chase D_1 using φ'. If $t_i[X'] \asymp t'_p[X']$ $(i \in [1,2])$, then change $t_i[Y']$ such that $t_i[Y'] \asymp t'_p[Y']$, as long as there exists no attribute $A \in Y'$ such that $t_i[A]$ is already a constant that does not match $t'_p[A]$. Moreover, if $t_1[X'] \doteq t_2[X']$ and $c' \leqslant c$, then change $t_1[Y'] \doteq t_2[Y']$ as long as there exists no attribute $A \in Y'$ such that $t_1[A]$ and $t_2[A]$ are already constants and $t_1[A] \neq t_2[A]$. Here $c = 1$ since φ is a CFD.

(b) For each attribute $B \in$ attr(R), if $t_i[B]$ $(i \in [1,2])$ is a variable after step (a), assign a distinct value from $dom(B)$ to $t_i[B]$; note that $dom(B)$ must be infinite in this case.

(c) If $D_1 \models \Sigma$ and $D_1 \not\models \varphi$, then return "yes".

The algorithm returns "no" if for all possible valuations to x_{A_i} and y_{A_i} for $i \in [1, k]$, it cannot instantiate D_1 such that $D_1 \models \Sigma$ but $D_1 \not\models \varphi$.

From these it follows that the algorithm returns "yes" iff $\Sigma \not\models \varphi$. In addition, similar to the proof of Proposition 3.2, it is easy to see that the algorithm is in PTIME in the sizes of Σ, φ, relation R_c (in the definition of $\doteq$), and the maximum cardinality of the k finite domains.

(2) A PTIME algorithm similar to the one given in the proof of (1) suffices to check whether $\Sigma \cup \{\neg\varphi\}$ is satisfiable. Here the algorithm operates on $c + 1$ tuples, as described in the proof of Theorem 4.2. Since Σ consists of CFDs only, the chase of the tuples using CFDs in Σ is straightforward. Since all the attributes in Σ or φ have an infinite domain, we no longer need to check valuations to those variables denoting attributes with a finite domain. One can verify that the algorithm is in PTIME. □

5. Concluding remarks

We have proposed CFD^{c}s and shown that CFD^{c}s have the following properties. (a) CFD^{c}s are able to express CFDs of [14], cardinality constraints, and domain-specific abbreviations and conventions in a uniform constraint formalism. (b) CFD^{c}s do not complicate the static analyses: the satisfiability and implication problems for CFD^{c}s have the same complexity bounds as their counterparts for CFDs.

One topic for future work is to develop a uniform constraint language to express CFD^{c}s and other extensions of CFDs, e.g., [9,16]. Such a language, however, comes at a price of higher complexity bounds: Proposition 3.2, for example, will no longer hold. This issue deserves a full treatment. Another topic is to revise the algorithms for computing a minimum cover of a set of CFDs [14], discovering CFDs [11,16] and for repairing data based on CFDs [13], by using CFD^{c}s instead of CFDs.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Fan and Ma are supported in part by EPSRC E029213/1. Fan is a Yangtze River Scholar at Harbin Institute of Technology.

References

[1] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[2] R. Ananthakrishna, S. Chaudhuri, V. Ganti, Eliminating fuzzy duplicates in data warehouses, in: VLDB, 2002.
[3] A. Arasu, S. Chaudhuri, R. Kaushik, Transformation-based framework for record matching, in: ICDE, 2008.
[4] M. Arenas, L.E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: PODS, 1999.
[5] M. Arenas, L.E. Bertossi, J. Chomicki, X. He, V. Raghavan, J. Spinrad, Scalar aggregation in inconsistent databases, Theoret. Comput. Sci. 296 (3) (2003) 405–434.
[6] M. Baudinet, J. Chomicki, P. Wolper, Constraint-generating dependencies, J. Comput. System Sci. 59 (1) (1999) 94–115.
[7] L. Bertossi, L. Bravo, E. Franconi, A. Lopatenko, The complexity and approximation of fixing numerical attributes in databases under integrity constraints, Inform. Systems 33 (4) (2008) 407–434.
[8] P.D. Bra, J. Paredaens, Conditional dependencies for horizontal decompositions, in: ICALP, 1983.
[9] L. Bravo, W. Fan, F. Geerts, S. Ma, Increasing the expressivity of conditional functional dependencies without extra complexity, in: ICDE, 2008.
[10] D. Calvanese, M. Lenzerini, On the interaction between ISA and cardinality constraints, in: ICDE, 1994.
[11] F. Chiang, R.J. Miller, Discovering data quality rules, in: VLDB, 2008.
[12] J. Chomicki, Consistent query answering: Five easy pieces, in: ICDT, 2007.
[13] G. Cong, W. Fan, F. Geerts, X. Jia, S. Ma, Improving data quality: Consistency and accuracy, in: VLDB, 2007.
[14] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, ACM Trans. on Database Systems 33 (1) (2008).
[15] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[16] L. Golab, H.J. Karloff, F. Korn, D. Srivastava, B. Yu, On generating near-optimal tableaux for conditional functional dependencies, in: VLDB, 2008.
[17] J. Grant, J. Minker, Normalization and axiomatization for numerical dependencies, Inform. and Control 65 (1) (1985) 1–17.
[18] P.C. Kanellakis, On the computational complexity of cardinality constraints in relational databases, Inform. Process. Lett. 11 (2) (1980) 98–101.
[19] M.J. Maher, Constrained dependencies, Theoret. Comput. Sci. 173 (1) (1997) 113–149.