Incorporating cardinality constraints and synonym rules into conditional functional dependencies

Citation for published version:
10.1016/j.ipl.2009.03.021

Digital Object Identifier (DOI):
10.1016/j.ipl.2009.03.021

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Information Processing Letters

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Incorporating cardinality constraints and synonym rules into conditional functional dependencies

Wenguang Chen\textsuperscript{a}, Wenfei Fan\textsuperscript{b,c,*}, Shuai Ma\textsuperscript{b}

\textsuperscript{a} Peking University, China
\textsuperscript{b} University of Edinburgh, UK
\textsuperscript{c} Bell Laboratories, USA

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:

\texttt{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:

$\phi_1$: ((\text{country}, \text{zip}) \rightarrow \text{street}, t_p^1), \quad \text{where } t_p^1 = (\text{UK}, \_ \_ _ _) \quad \phi_2$: (country $\rightarrow$ state, $t_p^2$) \quad \text{where } t_p^2 = (\text{UK} \parallel \text{N/A})$

Here $\phi_1$ 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, $\phi_2$ 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...
rule $\phi_1$. 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 $\phi_1$ and $\phi_2$. Indeed, these CFDs are specified with the pattern $\text{country} = \text{UK}$, 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 with a certain pattern does not exceed a predefined bound. As an 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 (domain-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$^s$, that is able to express cardinality constraints, synonym rules and patterns of semantically related values of CFD$^s$ in a uniform constraint formalism. For example, all constraints we have seen so far can be expressed as CFD$^s$.

2. We establish complexity bounds for the satisfiability problem and the implication problem associated with CFD$^s$. The satisfiability problem is to determine whether a set $\Sigma$ of CFD$^s$ has a nonempty model, i.e., whether the data quality rules in $\Sigma$ make sense. The implication problem is to decide whether a set $\Sigma$ of CFD$^s$ entails another CFD$^s$ $\phi$, i.e., whether the rule $\phi$ is redundant given the rules in $\Sigma$.

3. We identify special cases where the satisfiability and implication analyses of CFD$^s$ are in PTIME. That is, in these practical settings we are able to reason about CFD$^s$ efficiently.

We contend that CFD$^s$ 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$^s$, 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$^s$ 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

<table>
<thead>
<tr>
<th>FN</th>
<th>LN</th>
<th>street</th>
<th>city</th>
<th>state</th>
<th>country</th>
<th>zip</th>
<th>item</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>Joe</td>
<td>Brady</td>
<td>Mayfield</td>
<td>EDI</td>
<td>N/A</td>
<td>UK</td>
<td>EH4</td>
<td>8LE</td>
</tr>
<tr>
<td>t₂</td>
<td>Mark</td>
<td>Webber</td>
<td>Crichton</td>
<td>EDI</td>
<td>NY</td>
<td>United Kingdom</td>
<td>EH4</td>
<td>8LE</td>
</tr>
<tr>
<td>t₃</td>
<td>John</td>
<td>Hull</td>
<td>Queen</td>
<td>EDI</td>
<td>N/A</td>
<td>UK</td>
<td>EH4</td>
<td>8LE</td>
</tr>
<tr>
<td>t₄</td>
<td>William</td>
<td>Smith</td>
<td>5th Ave</td>
<td>NYC</td>
<td>NY</td>
<td>US</td>
<td>10016</td>
<td>book1</td>
</tr>
<tr>
<td>t₅</td>
<td>Bill</td>
<td>Smith</td>
<td>5th Ave</td>
<td>NYC</td>
<td>NY</td>
<td>US</td>
<td>10016</td>
<td>book2</td>
</tr>
<tr>
<td>t₆</td>
<td>Bill</td>
<td>Smith</td>
<td>5th Ave</td>
<td>NYC</td>
<td>NY</td>
<td>US</td>
<td>10016</td>
<td>book3</td>
</tr>
</tbody>
</table>

**Fig. 1.** An instance of the sale relation schema.
the impact of cardinality constraints on the satisfiability and implication analyses has been considered.

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

2. CFD’s: An extension of CFDs

Consider a relation schema R defined over a set of attributes, denoted by attr(R). For each attribute A ∈ 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’s.

CFD’s. A CFD ϕ defined on schema R is a triple R(X → Y, tp, c), where (1) X → Y is a standard FD, referred to as the FD embedded in ϕ; (2) tp is a tuple with attributes in X and Y, referred to as the pattern of ϕ, where for each A in X ∪ Y, tp[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, tp 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 tp[X], then t[Y] must observe the pattern tp[Y]. Furthermore, for all those tuples t such that t[X] has pattern tp[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 tp[X] pattern.

If A occurs in both X and Y, we use tp[A L] and tp[A R] to indicate its occurrence in X and Y, respectively. We separate the X and Y attributes in tp with ‘∥’, and denote X as LHS(ϕ) and Y as RHS(ϕ). We write ϕ as (X → Y, tp, c) when R is clear from the context.

**Example 2.1.** CFDs ϕ1 and ϕ2 of Example 1.1 can be expressed as CFD’s below, in which t_p^1 and t_p^2 are pattern tuples given in Example 1.1:

ϕ1 := ([country, zip] → street, t_p^1, 1),
ϕ2 := (country → state, t_p^2, 1).

The cardinality constraint described in Example 1.1 can also be written as a CFD ϕ3 := (fd, t_p^3, 2), where FD fd and pattern tuple t_p^3 are:

fd : fn, ln, street, city, state, country, zip, type → item,

t_p^3 = (._._._._._._. sale _._).

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

**Semantics of CFD’s.** To give the semantics of CFD’s, we first extend the equality relation and revise the match operator of [14].

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

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

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

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

**Matching operator.** We revise the binary operator ⊠ of [14] defined on constants and ‘_’ as follows: η_1 ⊠ η_2 if either (a) η_1 and η_2 are constants and η_1 = η_2, or (b) one of η_1, η_2 is ‘_’. The operator ⊠ extends to tuples, e.g., (a, b) ⊠ (c, d) but (a, b) ⊠ (c, e) if b ≠ e.

**Semantics.** Based on = and ⊠, we now give the semantics of CFD ϕ := R(X → Y, tp, c).

An instance D of schema R satisfies ϕ, denoted by D |= ϕ, iff for each tuple t in D, if t[X] ⊢ tp[X], then (1) t[Y] ⊠ tp[Y], and (2) |πY(σX∈D)| ⊣ c, i.e., for all tuples t in D such that t[X] ⊢ tp[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 ⊢ b.

Intuitively, ϕ is a constraint defined on the set of tuples D_ϕ := {t | t ∈ D, t[X] ⊢ tp[X]} such that (a) for each t ∈ D_ϕ, the pattern tp[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 |= of the equality relation. Note that ϕ is defined on the subset D_ϕ of D identified by tp[X], rather than on the entire D.

An instance of D satisfies a set Σ of CFD’s, denoted by D |= Σ, if D |= ϕ for each ϕ ∈ Σ.

**Example 2.2.** Assume that Rc consists of (”United Kingdom”, “UK”) and (”William”, “Bill”). Recall instance D_0 of Fig. 1 and CFD’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[country] = UK but t_2[state] ≠ N/A; (b) t_1, t_2 and t_3 violate ϕ_3 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 = Bill), all have type = sale, but they have three distinct items, beyond the bound 2.
Three special cases of CFDs are worth mentioning. (a) Traditional FDs are CFDs in which \( c = 1 \) and the pattern tuple consists of ‘_’ only. (b) CFDs of [14] are CFDs in which \( c \) is fixed to be 1. (c) Constant CFDs are CFDs 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 CFDs is the satisfiability problem.

The satisfiability problem for CFDs is to determine, given a set \( \Sigma \) of CFDs on a schema \( R \), whether or not there exists a nonempty instance \( D \) of \( R \) such that \( D \models \Sigma \).

Intuitively, the satisfiability problem is to decide whether a set of CFDs makes sense or not. When CFDs 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 CFDs’ subsume CFDs, the satisfiability problem for CFDs is at least as hard as for CFDs.

Example 3.1. Consider a schema \( R(A, B, C) \), and a set \( \Sigma_1 \) consisting of three CFDs defined on \( R \): \( \psi_1 = (A \rightarrow B) \), \( \psi_2 = (A \rightarrow B, \text{false} \mid b), 1 \), and \( \psi_3 = (C \rightarrow B, _1 \mid b'), 1 \). Then \( \Sigma_1 \) is not satisfiable. Indeed, for any nonempty instance \( D \) of \( R \) and any tuple \( t \) in \( D \), \( \psi_2 \) requires \( t[B] \) to be \( b' \) no matter what value \( t[C] \) is, whereas \( \psi_1 \) and \( \psi_3 \) force \( t[B] \) to be \( b \) no matter whether \( t[A] \) is true or false.

The intractability. Despite the increased expressive power, CFDs do not complicate the satisfiability analysis. Indeed, the satisfiability problem for CFDs 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 CFDs is NP-complete.

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

We show the upper bound by presenting an NP-algorithm that, given a set \( \Sigma \) of CFDs on a schema \( R \), checks whether \( \Sigma \) is satisfiable. Similar to CFDs [14], CFDs 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 \( \Sigma \) satisfiable.

Assume without loss of generality that \( \text{attr}(R) = \{A_1, \ldots, A_n\} \). For each \( i \in [1, n] \), define the active domain of \( A_i \) to be a set \( \text{dom}(A_i) \) consisting of all constants of \( \text{tp}(A_i) \) for all pattern tuples \( \text{tp} \) in \( \Sigma \), plus an extra distinct constant \( \text{dom}(A_i) \) if there exists one. Then it is easy to verify that \( \Sigma \) is satisfiable iff there exists a mapping \( \rho \) that assigns a value in \( \text{dom}(A_i) \) 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 \( t[A_i] \in \text{dom}(A_i) \) 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 \( x = y \), which can be done in PTIME in the sizes of \( \Sigma \) and \( R_c \), where \( R_c \) is the relation given in the definition of \( \models \). 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 CFDs with a finite domain. This motivates us to consider the following special case.

A set \( \Sigma \) of CFDs is said to be bounded by a constant \( k \) if at most \( k \) attributes in the CFDs of \( \Sigma \) have a finite domain. In particular, when \( k = 0 \), all CFDs in \( \Sigma \) are defined in terms of attributes with an infinite domain.

Bounded CFDs 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 \( \Sigma \) of CFDs is satisfiable if \( \Sigma \) is bounded by a constant \( k \).

Proof. When \( \Sigma \) is bounded by \( k \), we develop a PTIME algorithm to determine whether \( \Sigma \) 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 \( \models \) operator. Assume without loss of generality that \( \Sigma \) is defined on a schema \( R \), and only attributes \( A_i \) in CFDs of \( \Sigma \) 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 \( t[A_i] \) is a distinct variable \( x_A \) for each \( A_i \in \text{attr}(R) \). For all \( i \in [1, k] \), and for each value in \( \text{dom}(A_i) \) assigned to \( x_A \), the algorithm does the following.

(a) For each CFD \( \phi = R(X \rightarrow Y, t_p, c) \) in \( \Sigma \), when \( \phi \) does not appear in \( \text{attr}(R) \), assign a distinct value from \( \text{dom}(B) \) to \( x_B \), which does not appear in \( \Sigma \) and \( R_c \); note that \( \text{dom}(B) \) must be infinite in this case by the definition of \( t \).

(b) For each attribute \( B \in \text{attr}(R) \), if \( t[B] \) is still \( x_B \) after step (a), assign a distinct value from \( \text{dom}(B) \) to \( x_B \), which does not appear in \( \Sigma \) and \( R_c \); note that \( \text{dom}(B) \) 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 \( x_A \), for \( i \in [1, k] \), it cannot instantiate \( t \) such that \( t \models \Sigma \).

The algorithm is in \( O(\vert \Sigma \vert^2 R_c n^k) \) time, i.e., in PTIME when \( k \) is fixed, where \( \vert \Sigma \vert \) is the size of \( \Sigma \), \( |R_c| \) is the size of \( R_c \) (in the definition of \( \models \)), and \( m \) is the maximum cardinality of finite domains \( \text{dom}(A_i) \) for \( i \in [1, k] \).

We next show that the algorithm returns “yes” if and only if \( \Sigma \) is satisfiable.

If the algorithm returns “yes”, there exists a tuple \( t \) such that \( t \models \Sigma \). Thus \( \Sigma \) is satisfiable.

Conversely, if \( \Sigma \) is satisfiable, there exists a tuple \( t \) such that \( t \models \Sigma \). 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 \( t'[A] = \lambda A \) 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 \( t'[A] \models t[A] \). 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 \( \Sigma \), and thus have no impact on whether \( t' \) satisfies \( \Sigma \). 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's.

Consider a set \( \Sigma \) of CFD's and a single CFD \( \phi \) defined on the same schema \( R \). We say that \( \Sigma \) implies \( \phi \), denoted by \( \Sigma \models \phi \), iff for all instances \( D \) of \( R \), if \( D \models \Sigma \) then \( D \models \phi \). We consider without loss of generality satisfiable \( \Sigma \) only.

The implication problem for CFD's is to determine, given a set \( \Sigma \) of CFD's and a CFD \( \phi \) defined on the same schema, whether \( \Sigma \models \phi \).

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's of the form:

\[
\phi : R(\rightarrow X, t_p, c), \quad \phi' : R(\rightarrow Y, t_p, c')
\]

(a) \( \phi \models \phi' \) if \( c \leq c' \); and

(b) if \( \phi \) is a constant CFD, \( \phi \models \phi' \) even when \( c = 1 \) and \( c > c' \).

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

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's, along the same lines as its CFD counterpart (Theorem 4.3 in [14]).

In the rest of the section we consider a set \( \Sigma \) of CFD's and a CFD \( \phi = R(\rightarrow Y, t_p, c) \) such that \( c \) is bounded by a polynomial in the sizes of \( \Sigma \) and \( \phi \). This assumption is acceptable since in practice, \( c \) is typically fairly small.

Theorem 4.2. The implication problem for CFD's is coNP-complete.

Proof. The implication problem for constant CFDs is coNP-hard [14]. The lower bound carries over to CFD'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 \( \Sigma \not\models \phi \). The algorithm is based on a small model property: if \( \phi = R(\rightarrow X, t_p, c) \) and \( \Sigma \not\models \phi \), then there exists an instance \( D \) of \( R \) with at most \( c+1 \) tuples such that \( D \models \Sigma \) and \( D \not\models \phi \). That is, \( D \) consists of \( c+1 \) tuples \( t_1, \ldots, t_{c+1} \) such that for all \( i, j \in [1, c+1] \), \( t_i[X] \neq t_j[X] \) and \( t_i[X] = t_j[X] \), but either there exists \( i \in [1, c+1] \) such that \( t_i[Y] \neq t_p[Y] \), or for all \( i \neq j \), \( t_i[Y] \neq t_j[Y] \). Thus it suffices to consider instances \( D \) with \( c+1 \) tuples for deciding whether \( \Sigma \not\models \phi \).

We show that the problem is in coNP by presenting an NP algorithm for its complement, i.e., for deciding whether \( \Sigma \not\models \phi \). The implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2. We show that the implication problem is in coNP by Theorem 4.2.

Proposition 4.3. It is coNP-complete to decide, given CFD's \( \Sigma \) and \( \phi \), whether \( \Sigma \models \phi \) when \( \Sigma \cup \{ \phi \} \) 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 \( \Sigma \not\models \phi \), where 3SAT is NP-complete (cf. [15])). Consider an instance \( \phi = C_1 \land \cdots \land C_n \) of 3SAT, where all the variables in \( \phi \) are \( x_1, \ldots, x_m \), \( C_j \) is of the form \( y_j \lor y_j \lor y_j \), and moreover, for each \( i \in [1, 3] \), \( y_i \) is either \( x_{p_j} \) or \( \neg x_{p_j} \) for \( p_j \in [1, m] \); here we use \( x_{p_j} \) and \( \neg x_{p_j} \) to indicate the occurrence of a variable in literal \( i \) of clause \( C_j \). Given \( \phi \), we construct a relation schema \( R \), an empty relation \( R_n \), and a set \( \Sigma \cup \{ \phi \} \) of CFD's defined on \( R \), such that \( \phi \) is satisfiable iff \( \Sigma \not\models \phi \).

(1) We define schema \( R(C, V, X, V, Z) \), where \( dom(C) = \{1, \ldots, n\} \), \( dom(V) = \{b_1, b_2, b_3\} \), \( dom(X) = \{x_1, \ldots, x_m\} \), \( dom(Y) = \{y_1, \ldots, y_3\} \), which is the set of variables in \( \phi \), and moreover, both \( dom(V) \) and \( dom(Z) \) are integer. Intuitively, for each tuple \( t \in R \), \( t[C], t[V], t[X], t[V] \) and \( t[Z] \) specify a clause \( C \), a truth assignment \( \xi \), and the truth value of the variable and the truth value of \( C \) determined by \( \xi \).

(2) Let the set \( \Sigma \) be \( \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4 \).

(a) \( \Sigma_1 \) encodes the relationships among attributes \( C, V, X, V, Z \). For each variable in a clause \( C_j (1 \leq j \leq n) \) and each value \( \langle b_1 b_2 b_3 \rangle \) in \( dom(V) \), there is a CFD \( \phi \) in \( \Sigma_1 \).
Thus there are $3 \cdot 8 \text{CFD}^c$s for each clause $C_j$ in $\Sigma$, and in total, there are $24 \text{CFD}^c$s in $\Sigma$.

Each CFD$^c$ for clause $C_j = y_j \lor y_j \lor y_j$ is of the form of $R((C, Vc, X \rightarrow Vx), t_p, 1)$ such that $t_p(C) = j$, $t_p[Vc] = (b_1, b_2, b_3)$, and $t_p[X] = x_{pj}, (1 \leq i \leq 3)$. The value of $t_p[Vc]$ is decided by the value of $t_p[Vx]$ such that $t_p[Vx] = b_1$ if $y_j = x_{pj}$ and otherwise $t_p[Vx] = 1 - b_1$ if $y_j = x_{pj}$.

For example, if $C_j = x_{p1} \lor x_{p2} \lor x_{p3}$ such that $1 \leq p_1, p_2, p_3 \leq m$, then some possible pattern tuples are $(j, (010), x_{p1}, 0), (j, (010), x_{p2}, 0)$, and $(j, (010), x_{p3}, 0)$.

(b) $\Sigma_2$ 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 $\Sigma_2$: $\mu_{j,i,1} = 0$ and $\mu_{j,i,2} = 0$. 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 $(m - 3) \cdot n$ CFD$^c$s in $\Sigma_2$.

(c) $\Sigma_3$ 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 $h \in dom(Vc)$, $\mu_{h} = R((C, X \rightarrow Z), (j, x_i \in \{1, 0\}))$. Then no tuple $t$ satisfies $t[C] = j$ and $t[Z] = x_i$, since otherwise $\mu_{h}$ forces $t[Z] = 1$.

Taking 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 $\Sigma$ of CFD$^c$s and a CFD$^c$ $\varphi$, whether $\Sigma \models \varphi$ when $\Sigma \cup \{\varphi\}$ is bounded by a constant $k$ and one of the following conditions holds:

1. $\varphi$ is a CFD while $\Sigma$ is a set of CFD$^c$s; or
2. $\Sigma$ is a set of CFD$s$, $\varphi$ is a CFD$^c$ and $k = 0$, i.e., $\varphi$ (in $\Sigma$ or $\varphi$) has an infinite domain.

Proof. Observe that $\Sigma \models \varphi$ iff there exists a nonempty instance $D$ of the schema $R$ on which both $\Sigma$ and $\varphi$ are defined, such that $D \models \Sigma \cup \{\varphi\}$. Thus it suffices to develop a PTIME algorithm to check the satisfaction of $\Sigma \cup \{\varphi\}$. Assume without loss of generality that $\Sigma$ is defined on a schema $R$, and only attributes $A_i$ in CFD$^c$s of $\Sigma$ 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 $x_A$ and $y_A$ with values in $dom(A_i)$, the algorithm does the following.

(a) For each CFD$^c$ $\varphi = R(X' \rightarrow Y', t_p', c')$ in $\Sigma$, chase $D_1$ using $\varphi$. If $t_i[X'] \not\equiv t_p'[X']$ (i.e., $i \in [1, 2]$), then change $t_i[Y']$ such that $t_i[Y'] \equiv t_p'[Y']$, 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 $t_i[X'] \equiv t_p'[X']$ and $c' = c$, then change $t_i[Y'] \equiv t_p'[Y']$ as long as there exists no attribute $A \in Y'$ such that $t_i[A]$ and $t_p[A]$ are already constants and $t_i[A] \not\equiv t_p[A]$. Here $c = 1$ since $\varphi$ is a CFD.

(b) For each attribute $B \in attr(R)$, if $t_i[B]$ (i.e., $i \in [1, 2]$) is a variable after step (a), assign a distinct value from $dom(B)$ to $t_1[B]$; 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 $\Sigma$, $\varphi$, relation $R_C$ (in the definition of $\equiv$), 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 $\Sigma$ consists of CFDs only, the chase of the tuples using CFDs in $\Sigma$ is straightforward. Since all the attributes in $\Sigma$ or $\varphi$ 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.

References