The Parameterized Complexity of Dependency Detection in Relational Databases

Thomas Bläsius[1], Tobias Friedrich[2], and Martin Schirneck[3]

1 **Hasso Plattner Institute, Potsdam, Germany**
thomas.blaesius@hpi.de
2 **Hasso Plattner Institute, Potsdam, Germany**
tobias.friedrich@hpi.de
3 **Hasso Plattner Institute, Potsdam, Germany**
martin.schirneck@hpi.de

Abstract

We study the parameterized complexity of classical problems that arise in the profiling of relational data. Namely, we characterize the complexity of detecting *unique column combinations* (candidate keys), *functional dependencies*, and *inclusion dependencies* with the solution size as parameter. While the discovery of uniques and functional dependencies, respectively, turns out to be $W[2]$-complete, the detection of inclusion dependencies is one of the first natural problems proven to be complete for the class $W[3]$. As a side effect, our reductions give insights into the complexity of enumerating all minimal unique column combinations or functional dependencies.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases parameterized complexity, unique column combination, functional dependency, inclusion dependency, profiling relational data

Digital Object Identifier 10.4230/LIPIcs.IPEC.2016.6

1 Introduction

Data profiling is the process of gathering metadata from a given database, which in turn facilitates various tasks such as data cleansing, normalization and integration as well as query optimization. A common problem in data profiling is the detection of different types of dependencies between pieces of data, most notably unique column combinations, functional dependencies, and inclusion dependencies. Due to their practical relevance, these three problems have received much attention, which lead to numerous detection as well as enumeration algorithms, see e.g. the survey by Abedjan, Golab and Naumann [1]. Despite the fact that these algorithms perform well in practice, there are usually no theoretical performance guarantees. This is not very surprising as all three problems are known to be intractable: finding a minimum unique column combination is NP-complete [3] and cannot be approximated within a factor of $1/4 \log n$ (under reasonable complexity assumptions) [2], finding a minimum functional dependency is also NP-complete [7] and finding a maximum inclusion dependency is NP-complete even for restricted cases [14].

One approach to overcome these difficulties is to exploit properties that are usually observed in realistic data to design algorithms that guarantee a polynomial run time in case these features are present in the problem instance. Consider for example the histograms in Figure 1, showing the size distribution of minimal unique column combinations, minimal functional dependencies, and maximal inclusion dependencies in the MusicBrainz database [18]. Usually the majority of functional dependencies (as well as unique column

11th International Symposium on Parameterized and Exact Computation (IPEC 2016).
Editors: Jiong Guo and Danny Hermelin; Article No. 6; pp. 6:1–6:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

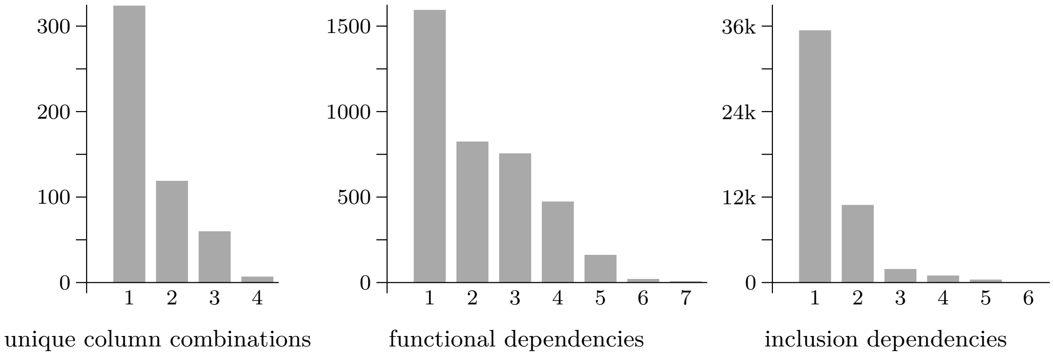


Figure 1 The number of minimal unique column combinations, minimal functional dependencies and maximal inclusion dependencies for given solution sizes in the MusicBrainz database.

combinations and inclusion dependencies) are rather small. Beside surrogate keys, giving rise to multiple functional dependencies of size 1, natural causalities also lead to small functional dependencies. For example, the name of an event together with the year in which it starts determines the year in which it ends, implying a functional dependency of size 2. Note that the starting year alone is not enough to infer this information. The name of the action, however, seems to indicate whether the event ends in the starting year or the following one.

Although the size k of the minimum functional dependency can in principle be (almost) as large as the total number of attributes, it appears to be a reasonable assumption that k is significantly smaller. It is thus very natural to ask whether the problem of finding a minimum functional dependency is *fixed-parameter tractable* (FPT) with respect to k, i.e., whether it can be solved in time $O(f(k) \cdot p(n))$, where p is a polynomial in the input size n, while f is an arbitrary function in the parameter k, but not in n. Note that the running time of an FPT-algorithm in general can still be superpolynomial. However, when assuming the parameter k to be bounded by a constant, one obtains a polynomial running time, in $O(p(n))$, whose order of growth does not depend on k. Hence, one can think of parameterized complexity as being a more fine-grained approach to complexity theory.

Parameterized complexity has been a great success in the design and analysis of algorithms [5, 11]. Nevertheless, its techniques have rarely been employed in the context of database theory so far. A notable exception is the complexity of database queries. Papadimitriou and Yannakakis [17] considered this problem for different query languages using the size of the query or alternatively the number of variables as the parameter. They showed that presumably none of the variants admits an FPT-algorithm, as the resulting problems are at least $W[1]$-hard (some are actually $W[t]$-hard for any positive integer t, $W[\text{SAT}]$-hard or even $W[\text{P}]$-hard). For further results on the parameterized complexity of database queries see the survey by Grohe [13]. Besides queries, we are not aware of any algorithmic database problems that have been considered through the lens of parameterized complexity.

Our Contribution and Outline. We show that detecting minimum unique column combinations and minimum functional dependencies are both $W[2]$-complete problems. Also, we prove that finding maximum inclusion dependencies is $W[3]$-complete. Thereby we completely settle the parameterized complexity of these problems with the solution size as parameter. We would like to point out that the completeness for the class $W[3]$ of a well-studied problem like the discovery of inclusion dependencies is quite surprising as natural problems are rarely $W[3]$-complete. In fact, besides a result by Chen and Zhang [4] related to supply chain management, we are not aware of *any* natural $W[t]$-complete problem for $t > 2$.

In Section 2 we give basic definitions and formal problem statements. In Section 3, we examine the detection of minimum unique column combinations as well as minimum functional dependencies. For the latter, we actually consider two variants, one for which the right hand side of the functional dependency is fixed and one in which it is variable. We show that all three problems are $W[2]$-complete. As a byproduct, our reductions (involving the problem Hitting Set) have certain implications on the computational hardness of enumerating all unique column combinations or functional dependencies of a given relation. See the end of Section 3 for more details. In Section 4 we show that finding minimum inclusion dependencies for a pair of relations is $W[3]$-complete. We also show that the problem remains $W[3]$-complete if both relations are defined over the same schema together with a fixed mapping between the columns of the tables. In Section 5, we conclude this paper by discussing alternative parameter choices as well as possible future research in general.

2 Notation and Problems

2.1 Parameterized Complexity

For an instance I of a decision problem and a parameter $k \in \mathbb{N}^+$, the pair (I, k) is an instance of the corresponding *parameterized problem*. The running time of an algorithm is then considered not only in terms of the input size $|I|$ but also in terms of k. A parameterized problem is *fixed-parameter tractable*, i.e., it belongs to the complexity class FPT, if a given instance can be solved in time $O(f(k) \cdot p(|I|))$, where p is a polynomial while f is an arbitrary computable function. We then also say that the algorithm runs in FPT-*time*.

Let P and P' be two parameterized problems. A *parameterized reduction* from P to P' is an algorithm running in FPT-time that maps an instance (I, k) of P to an equivalent instance (I', k') of P' such that the parameter k' depends only on the value of k (and not on $|I|$). Note that an (hypothetical) FPT-algorithm for P' would also yield an FPT-algorithm for P via this reduction. Hence, considering their parameterized complexity, P is at most as hard as P', which we denote by $P \leq_{\text{FPT}} P'$. If conversely $P' \leq_{\text{FPT}} P$ also holds, we say that P and P' are FPT-*equivalent*.

The parameterized reduction leads to a hierarchy of complexity classes, the so-called *W-hierarchy*, by specifying a *complete* problem for each class. To define the desired family of problems, we employ Boolean formulas in propositional logic. Let φ be such a formula. A satisfying truth assignment for φ has *Hamming weight* k if exactly k variables are set to true in this assignment; we also call the set of these k variables a *solution* for φ. The formula φ is *t-normalized* if it can be written as a conjunction of disjunctions of conjunctions of disjunctions (and so on) of literals with $t-1$ alternations between conjunction and disjunction. Observe that a Boolean formula is 2-normalized if it is in conjunctive normal form (CNF) and 3-normalized if it is a conjunction of subformulas in disjunctive normal form (DNF).

The problem Weighted t-normalized Satisfiability is to decide for a given t-normalized formula φ and a positive integer k whether φ has a weight k satisfying assignment; here k serves as the parameter. For any $t \geq 1$, a parameterized problem P is said to be in the complexity class $W[t]$ in case $P \leq_{\text{FPT}}$ Weighted t-normalized Satisfiability.[1]

The classes FPT $\subseteq W[1] \subseteq W[2] \subseteq \ldots$ form an ascending hierarchy and all inclusion are assumed to be proper, which is however still unproven [11]. The higher a problem ranks in the W-hierarchy the lower we consider the chances of finding an FPT-algorithm to solve it.

[1] We tacitly avoid the classical definition of $W[t]$ via weft-t-depth-d-families of decision circuits. This is justified by the Normalization Theorem by Downey and Fellows [9, 10].

2.2 Dependencies in Relational Databases

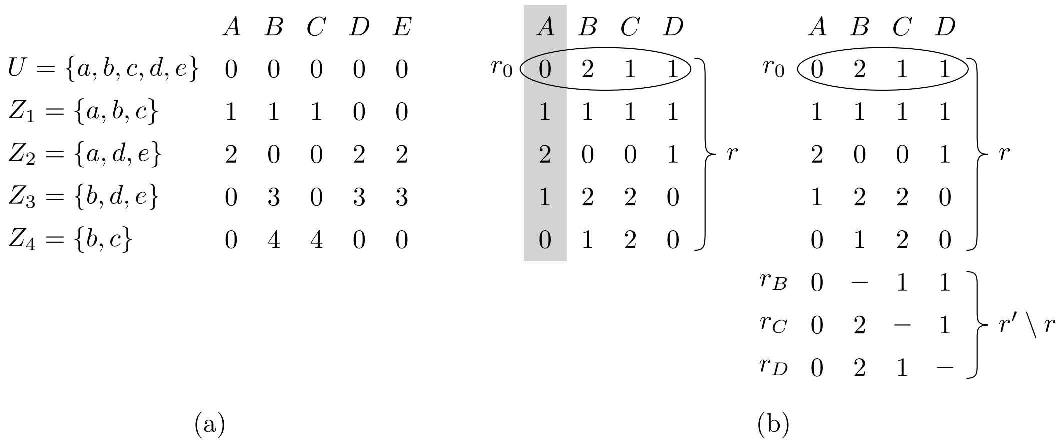
If not explicitly stated otherwise, notation regarding relational databases follows the survey by Abedjan et al. [1]. We let R and S be *relational schemata*, i.e., sets of *columns*; each column is associated with a set of admissible values. Symbols X, Y refer to sets of columns and symbols A, B refer to a single column, an *attribute.* We denote with r_i, r_j tuples whose entries, respectively, are indexed by some schema R and, for any subset $X \subseteq R$ of columns, we let $r_i[X]$ denote the sub-tuple of r_i consisting only of the entries indexed by X. In particular, $r_i[A]$ denotes the *value* of attribute A in r_i. A set r of such tuples is an *instance* of schema R if, for any $r_i \in r$ and $A \in R$, value $r_i[A]$ is admissible for attribute A. Instances of schematas are called *relations* or *relational databases* (over the corresponding schema). With $r[X]$ we denote the collection of all sub-tuples $r_i[X]$ for $r_i \in r$.

Let r be an instance of schema R. A collection $X \subseteq R$ of columns is called a *unique column combination* or *unique* if, for any two distinct tuples $r_i \neq r_j$ in r, we have $r_i[X] \neq r_j[X]$. So the combination of values for X fully identifies a tuple of relation r. Otherwise, X is called a *non-unique.* The *size* of a unique X is the cardinality $|X|$. Clearly, any superset of a unique is unique and any subset of a non-unique is again non-unique. The problem Unique is to decide for a given relational database r and a positive integer k whether r has a unique column combination of size at most k. Unique is known to be NP-complete [3].

A *functional dependency* (FD) over a schema R is an expression of the form $X \to A$ for some set $X \subseteq R$ of columns and an attribute $A \in R$. The set X is called the *left-hand side* (LHS) of the dependency and attribute A the *right-hand side* (RHS). A functional dependency $X \to A$ *holds* in an instance r (of schema R) if any pair of tuples that agree on X also agree on A, i.e., if $r_i[X] = r_j[X]$ implies $r_i[A] = r_j[A]$ for any two tuples $r_i, r_j \in r$. Otherwise, the FD is said to *fail* in r. A functional dependency is *non-trivial* if $A \notin X$ ($X \to A$ evidently holds if $A \in X$). The *size* of an FD is the cardinality of its LHS. The problem FD is to decide for a given relational database r and a positive integer k whether there is a non-trivial functional dependency of size at most k that holds in r. The problem $\text{FD}_{\textit{fixed}}$ is to decide for a given attribute $A \in R$, whether there is such a functional dependency with right-hand side A. The restricted variant $\text{FD}_{\textit{fixed}}$ is known to be NP-complete [7].

At last we define inclusions between columns among different relations. Let r be an instance of schema R and s be an instance of S. For some $X \subseteq R$, let $\sigma \colon X \to S$ be an injective map. Then the pair (X, σ) is an *inclusion dependency* (IND) if, for each tuple $r_i \in r$, there exists a tuple $s_j \in s$ such that $r_i[A] = s_j[\sigma(A)]$ for every $A \in X$, i.e., $r[X] \subseteq s[\sigma(X)]$. If the map σ is given in the input, we simply say that X is the inclusion dependency. The *size* of an inclusion dependency is $|X|$. The problem IND is to decide for two relations r and s (over schemata R and S, respectively) and a positive integer k whether there is an inclusion dependency (X, σ) of size at least k. In case of $R = S$ and σ being the identity mapping over R, the problem $\text{IND}_{\textit{fixed}}$ is to decide whether there is an inclusion dependency X of size at least k. Detecting an inclusion dependency in a relational database is NP-complete [14].

Note that the decision problems defined above do not depend upon asking for a solution of size at most/at least k as opposed to one of size exactly k. A functional dependency stays valid when adding arbitrary additional columns to the LHS. The same holds for unique column combinations. Conversely, if a pair of relations admits an inclusion dependency of size at least k, one can obtain one comprising exactly k columns by removing dispensable attributes. In this paper, we consider all the decision problems to be parameterized by the size of the respective solution. Thus, Unique, FD, $\text{FD}_{\textit{fixed}}$, IND and $\text{IND}_{\textit{fixed}}$ refer to the corresponding parameterized problems with parameter k.



	A	B	C	D	E
$U = \{a,b,c,d,e\}$	0	0	0	0	0
$Z_1 = \{a,b,c\}$	1	1	1	0	0
$Z_2 = \{a,d,e\}$	2	0	0	2	2
$Z_3 = \{b,d,e\}$	0	3	0	3	3
$Z_4 = \{b,c\}$	0	4	4	0	0

(a)

	A	B	C	D	
r_0	0	2	1	1	r
	1	1	1	1	
	2	0	0	1	
	1	2	2	0	
	0	1	2	0	

	A	B	C	D	
r_0	0	2	1	1	r
	1	1	1	1	
	2	0	0	1	
	1	2	2	0	
	0	1	2	0	
r_B	0	–	1	1	$r' \setminus r$
r_C	0	2	–	1	
r_D	0	2	1	–	

(b)

Figure 2 (a) An instance of Hitting Set and its equivalent instance of Unique. (b) An instance r of problem FD$_{\mathit{fixed}}$ with fixed RHS A and the resulting instance r' of problem FD. Note that the functional dependency $AB \to D$ holds in r but not in r'.

3 Unique Column Combinations and Functional Dependencies

It is a well-known phenomenon that throughout the fields of database design and data profiling theoreticians as well as practitioners are frequently confronted with the task of finding an inclusion-minimal collection of items that has a non-empty intersection with each member of a prescribed family of sets [1, 6, 16]. Thus, they aim to solve instances of the so-called Hitting Set problem. In this section we show that this encounter is somewhat inevitable in the sense that detecting uniques or functional dependencies is both *exactly* as hard as finding a hitting set in terms of parameterized complexity.

Hitting Set is formally defined as follows. For a finite system of subsets $\mathcal{Z} \subseteq \mathcal{P}(U)$ of some finite ground set U, a set $H \subseteq U$ is called *hitting set* iff for all $Z \in \mathcal{Z}$, $H \cap Z \neq \emptyset$. The problem Hitting Set is to decide for a positive integer k whether there is a hitting set H with $|H| \leq k$. Hitting Set is NP-complete [15] and $W[2]$-complete with respect to k [11]. Hence, we can utilize it to show the $W[2]$-completeness of the dependency problems at hand.

More precisely, in this section we establish a (seemingly ascending) chain of problems via parameterized reductions. This chain consists of problems Hitting Set, Unique, FD$_{\mathit{fixed}}$, FD and Weighted 2-normalized Satisfiability in that order. As the first and last problem are both $W[2]$-complete and thus FPT-equivalent, this in fact proves equivalence (and hence $W[2]$-completeness) for the other problems as well. Due to space constraints, we only sketch key ideas for the first three reductions (Hitting Set to Unique, Unique to FD$_{\mathit{fixed}}$ and FD$_{\mathit{fixed}}$ to FD).

▶ **Lemma 1.** Hitting Set $\leq_{\text{FPT}}$ Unique $\leq_{\text{FPT}}$ FD$_{\mathit{fixed}}$ $\leq_{\text{FPT}}$ FD.

Proof (sketch). The first reduction regarding Hitting Set and Unique is a straightforward translation of the sets to hit into tuples of a relational database, cf. Figure 2.(a). For the second reduction, the main idea is to add an extra column serving as a tuple ID and to subsequently show that a column combination is unique just in case it is the LHS of a functional dependency pointing to this ID. The last reduction is established by adding to a given relation copies of an already present tuple, "ruling out" a different entry in each copy, see Figure 2.(b). This then invalidates functional dependency with unwanted RHS. ◀

In the next lemma we prove that every instance of FD can be expressed by an equivalent

Boolean CNF-formula, establishing the reduction from FD to Weighted 2-normalized Satisfiability. This is the main result of this section.

▶ **Lemma 2.** FD $\leq_{\text{FPT}}$ Weighted 2-normalized Satisfiability.

Proof. Given a relation r (over some schema R), we derive a propositional formula that has a satisfying truth assignment of weight $k+1$ if and only if there is a non-trivial functional dependency of size k that holds in r. The formula will be in CNF and hence 2-normalized. We use two types of variables distinguished by their semantic purpose, namely, the elements of $\text{Var}_R = \{x_A \mid A \in R\}$ and $\text{Var}'_R = \{x'_A \mid A \in R\}$. A variable from Var_R being set to true denotes that the corresponding attribute appears on the LHS of the FD; for Var'_R, this denotes that the attribute is the RHS. Consequently, we want to ensure that any satisfying assignment chooses exactly one variable from Var'_R while the corresponding variable in Var_R is not chosen. We achieve this as follows. First, define some clause

$$c_R = \bigvee_{x'_A \in \text{Var}'_R} x'_A.$$

Then, for any two distinct attributes $A \neq B$, the clause

$$c_{A,B} = \neg x'_A \vee \neg x'_B.$$

Finally, we set, for every $A \in R$,

$$c_A = \neg x'_A \vee \neg x_A.$$

We note that the clauses $c_{A,B}$ and c_A are only added to smooth the analysis, the correctness of the reduction does not depend upon their presence. Clause c_R however is essential. Now, for any possible RHS $A \in R$ and any two tuples $r_i, r_j \in r$ with $r_i[A] \neq r_j[A]$, let

$$c_{A,r_i,r_j} = \neg x'_A \vee \bigvee_{\substack{B \in R \setminus A \\ r_i[B] \neq r_j[B]}} x_B.$$

That is, clause c_{A,r_i,r_j} contains the negative literal of the variable from Var'_R corresponding to the RHS and the positive literal of any variable in Var_R that corresponds to *another* attribute on which r_i and r_j disagree. Intuitively speaking, clause c_{A,r_i,r_j} states that if A is the RHS of a non-trivial FD holding in r, then the LHS has to contain at least one of the attributes $B \neq A$ such that $r_i[B] \neq r_j[B]$. We assemble the following Boolean formulas:

$$\varphi_{\text{RHS}} = c_R \wedge \bigwedge_{\substack{A,B \in R \\ A \neq B}} c_{A,B} \wedge \bigwedge_{A \in R} c_A;$$

as well as, for each possible RHS $A \in R$,

$$\varphi_A = \bigwedge_{\substack{r_i, r_j \in r \\ r_i[A] \neq r_j[A]}} c_{A,r_i,r_j}$$

and their conjunction

$$\varphi_{\text{LHS}} = \bigwedge_{A \in R} \varphi_A.$$

At last, we take $\varphi_{\text{FD}} = \varphi_{\text{LHS}} \wedge \varphi_{\text{RHS}}$ as the result of the reduction. In total, φ_{FD} has at most $|R|^2 + |R|\,|r|^2$ clauses with at most $|R|$ literals each and a representation of φ_{FD} is computable in time polynomial in the input size.

Regarding the correctness of this reduction, recall that we claimed φ_{FD} to have a weight $k+1$ satisfying assignment just in case there is a non-trivial functional dependency of size k in r. First assume we are given a satisfying assignment for subformula φ_{RHS}. This ensures that exactly one variable $x'_A \in \text{Var}'_R$ is set to TRUE, which uniquely determines an attribute A. The variables in Var_R that are set to TRUE determine a set X, i.e., $B \in X$ iff x_B is set to TRUE. Note that $A \notin X$ is enforced by clause c_A. Thus, a satisfying assignment for φ_{RHS} defines a non-trivial functional dependency $X \to A$. We show that this FD holds in r if and only if the assignment additionally fulfills subformula φ_{LHS}.

Suppose that $X \to A$ holds in r. Then all variables in Var'_R are set to FALSE, except x'_A, automatically satisfying clauses c_{B,r_i,r_j} for all attributes $B \neq A$. It remains to show that c_{A,r_i,r_j} is satisfied for every pair of tuples $r_i, r_j \in r$ with $r_i[A] \neq r_j[A]$. Since $X \to A$ holds, X includes, for every such pair, an attribute B such that $r_i[B] \neq r_j[B]$. Clause c_{A,r_i,r_j} in turn comprises the literal x_B, which is satisfied by above assignment. Conversely, assume $X \to A$ fails in r. Then there is a pair of tuples $r_i, r_j \in r$ such that $r_i[A] \neq r_j[A]$ but $r_i[X] = r_j[X]$. The clause c_{A,r_i,r_j} does not contain any variables x_B such that $B \in X$. As a result, all literals in c_{A,r_i,r_j} for variables from Var_R evaluate to FALSE. Literal $\neg x'_A$, however, is FALSE as well as A is the RHS. ◀

The reductions in Lemma 1 and 2, and the fact that HITTING SET and WEIGHTED 2-NORMALIZED SATISFIABILITY are both $W[2]$-complete yield the following theorem.

▶ **Theorem 3.** *The problems* UNIQUE, $\text{FD}_{\textit{fixed}}$ *and* FD *are* $W[2]$*-complete.*

We point out that our reductions have some further implications beyond the scope of parameterized complexity. Our reduction from $\text{FD}_{\textit{fixed}}$ to FD, is actually a polynomial-time reduction and thus proves that FD is NP-hard. As the problem is also trivially contained in NP, it is in fact NP-complete. This is not very surprising, but has only been proven for the restricted case $\text{FD}_{\textit{fixed}}$ before. More importantly, the parameterized reduction from HITTING SET to UNIQUE, also runs in polynomial time and establishes a one-to-one correspondence between inclusion-wise minimal hitting sets and unique column combinations. Thus, *enumerating* all uniques is at least as hard as enumerating all hitting sets. Using the techniques presented in this section, it is not hard to extend this observation to the task of enumerating all FDs with a fixed RHS or all FDs in general. Finding all hitting sets is also known by the name TRANSVERSAL ENUMERATION for hypergraphs and is notoriously difficult [12]. Up until now, there is no output-polynomial algorithm known for this problem. Hence, it is quite astonishing that the enumeration problems arising in data profiling can be solved in reasonable time on practical data sets [1]. Regarding the opposite direction, we would like to mention that there is a straight-forward polynomial reduction from $\text{FD}_{\textit{fixed}}$ to HITTING SET that additionally shows that enumerating FDs with fixed RHS or (and thus uniques) is also *at most* as hard as enumerating hitting sets. It is still unknown, however, whether this holds for arbitrary functional dependencies as well.

4 Inclusion Dependencies

In this section, we identify the detection of inclusion dependencies as one of the first natural problems to be complete for the parameterized class $W[3]$. More precisely, we show that both IND and $\text{IND}_{\textit{fixed}}$ are FPT-equivalent to a $W[3]$-complete parameterized version of the satisfiability problem for certain Boolean formulas. Recall that a propositional formula is 3-normalized if it is a conjunction of disjunctions of conjunctions of literals. A formula is *antimonotone* if it only contains negative literals as, e.g., in the following expression

$$((\neg a \wedge \neg b) \vee (\neg c \wedge \neg d)) \wedge ((\neg a \wedge \neg c) \vee (\neg b \wedge \neg d)).$$

It is antimonotone, 3-normalized and admits a satisfying assignments of Hamming weight 0 and 1, but none of larger weight. The problem Weighted Antimonotone 3-normalized Satisfiability (WA3NS) is to decide for a 3-normalized antimonotone formula φ and a positive integer k whether φ has a weight k satisfying assignment. Although it seems to be a unreasonably restricted case of Weighted 3-normalized Satisfiability, WA3NS is $W[3]$-complete in its own right when parameterized by k [8, 9]. As all literals in WA3NS are negative, every subset of a solution is a solution. Thus, asking for a solution of size exactly k is again equivalent to asking for a solution of size at least k.

4.1 IND is in W[3]

In this subsection, we show that both variants of the IND problem are members of the parameterized class $W[3]$. As a first step, we reduce the special case IND$_{\textit{fixed}}$ to the (seemingly) more general problem IND.

▶ **Lemma 4.** IND$_{\textit{fixed}}$ $\leq_{\text{FPT}}$ IND.

Proof. Let r and s be two relations over the same schema R forming an instance of problem IND$_{\textit{fixed}}$. We introduce a new tuple $t^- = (-_A)_{A \in R}$, where each "$-_A$" is a unique symbol not used anywhere in the relations r or s. Note that (r, s) has an inclusion dependency of size k with the identity as the fixed mapping if and only if $(r \cup \{t^-\}, s \cup \{t^-\})$ has one. It is easy to see that $(r \cup \{t^-\}, s \cup \{t^-\})$ interpreted as an instance of IND (now without prescribed mapping) has an inclusion dependency (X, σ) if and only if σ is the identity and X is an inclusion dependency for (r, s), which implies the claim. ◀

To reduce IND to WA3NS we construct from the two relations an antimonotone formula which has a weight k satisfying assignment if and only if the relations have an inclusion dependency of the same size. For this, we use a correspondence between *pairs* of attributes of the relational schemata and Boolean variables.

▶ **Theorem 5.** IND $\leq_{\text{FPT}}$ WA3NS.

Proof. Let $R = \{A_1, \ldots, A_{|R|}\}$ and $S = \{B_1, \ldots, B_{|S|}\}$ be two schemata. We introduce a Boolean variable $x_{m,n}$ for each pair of attributes $A_m \in R$ and $B_n \in S$. We let Var_P denote the set of variables corresponding to a collection $P \subseteq R \times S$ of such pairs. Consider subsets $X \subseteq R$ and $Y \subseteq S$ together with a bijection $\sigma \colon X \to Y$. From this we can construct a truth assignment by setting variable $x_{m,n}$ to true iff $A_m \in X$ and $\sigma(A_m) = B_n$ (implying $B_n \in Y$). The resulting assignment has weight $|X|$ and the collection of all possible configurations (X, σ) is uniquely described by $\text{Var}_{R \times S}$ and the truth assignments obtained this way. Moreover, these assignments all satisfy the following Boolean formula.

$$\varphi_{\text{map}} = \left(\bigwedge_{m=1}^{|R|} \bigwedge_{n=1}^{|S|-1} \bigwedge_{n'>n}^{|S|} (\neg x_{m,n} \vee \neg x_{m,n'}) \right) \wedge \left(\bigwedge_{n=1}^{|S|} \bigwedge_{m=1}^{|R|-1} \bigwedge_{m'>m}^{|R|} (\neg x_{m,n} \vee \neg x_{m',n}) \right).$$

The first half of φ_{map} states that, for every pair of variables $x_{m,n}$ and $x_{m,n'}$ such that $n \neq n'$, at most one of them is set to true; the second half is satisfied if the same holds for all pairs $x_{m,n}$

and $x_{m',n}$ such that $m \neq m'$. Conversely, given a satisfying assignment for φ_{map}, we obtain sets $X = \{A_m \mid \exists\, 1 \leq n \leq |S| \colon x_{m,n} = \text{TRUE}\}$ and $Y = \{B_n \mid \exists\, 1 \leq m \leq |R| \colon x_{m,n} = \text{TRUE}\}$ as well as a bijection $\sigma \colon X \to Y$ by setting $\sigma(A_m) = B_n$ iff $x_{m,n}$ is TRUE. So φ_{map} is *exactly* fulfilled by the assignments described above.

We now formalize the requirement that (X, σ), for some set $X \subseteq R$, actually is an inclusion dependency. First, assume that the relations r and s consist of a single tuple r_i and s_j, respectively. We say a pair (A_m, B_n) is *forbidden* for r_i and s_j if $r_i[A_m] \neq s_j[B_n]$. Let $F_{i,j}$ be the set of all forbidden pairs. Then (X, σ) is an inclusion dependency if $x_{m,n}$ is set to FALSE for all pairs $(A_m, B_n) \in F_{i,j}$. In terms of Boolean formulas, this is represented as

$$\varphi_{i,j} = \bigwedge_{x \in \text{Var}_{F_{i,j}}} \neg x.$$

It follows that (X, σ) is an inclusion dependency if and only if the corresponding variable assignment satisfies both φ_{map} and $\varphi_{i,j}$.

Now suppose s has multiple tuples (while r is still considered to have only one). (X, σ) is an inclusion dependency for (r, s) just in case it is an inclusion dependency for *at least one* instance $(r, \{s_j\})$, $1 \leq j \leq |s|$. If also r has more than one tuple, then (X, σ) is an inclusion dependency for (r, s) if it is one in *each* instance $(\{r_i\}, s)$, $1 \leq i \leq |r|$. Thus, we obtain an inclusion dependency if and only if φ_{map} and the formula

$$\varphi = \bigwedge_{i=1}^{|r|} \bigvee_{j=1}^{|s|} \varphi_{i,j}$$

are simultaneously satisfied by the assignment corresponding to (X, σ).

The formula $\varphi \wedge \varphi_{\text{map}}$ is antimonotone and 3-normalized and a representation can be computed in polynomial time. Moreover, by the above observation that any solution for the sub-formula φ_{map} that corresponds to (X, σ) has size $|X|$, the reduction preserves the parameter. ◀

▶ **Corollary 6.** *$\text{IND}_{\textit{fixed}}$ and IND are both in class $W[3]$.*

4.2 IND is W[3]-hard

In the remainder of this section, we argue that the existence of weight k satisfying assignments for 3-normalized antimonotone formulas can be decided by solving $\text{IND}_{\textit{fixed}}$ instances. As a result, we prove that both variants of problem IND are hard for the class $W[3]$. For this reduction we make use of indicator functions, which we will define in a moment. First, we would like to point out that in the following we interpret propositional formulas φ over n variables as functions $f_\varphi \colon \{0,1\}^n \to \{0,1\}$ in the obvious way. For an instance (r, s) of $\text{IND}_{\textit{fixed}}$, we encode any subset $X \subseteq R$ using its characteristic vector (of length $|R|$). Then we define the *indicator function* $f_{(r,s)} \colon \{0,1\}^{|R|} \to \{0,1\}$, where $f_{(r,s)}(X) = 1$ iff X is an inclusion dependency. We claim that for any antimonotone and 3-normalized formula φ, there is an instance (r, s) of $\text{IND}_{\textit{fixed}}$ computable in FPT-time such that $f_\varphi = f_{(r,s)}$. This clearly gives an FPT-reduction from WA3NS to $\text{IND}_{\textit{fixed}}$. The remaining lemmas are dedicated to proving this claim.

Recall that an antimonotone, 3-normalized formula is a conjunction of antimonotone sub-formulas in DNF. Thus, the top level connective is a conjunction. Next we show how to model this connective in terms of relational databases.

▶ **Lemma 7.** *Let $(r^{(1)}, s^{(1)})$ and $(r^{(2)}, s^{(2)})$ be two instances for the problem* $\text{IND}_{\textit{fixed}}$ *(all relations are over the same schema R) with indicator functions $f^{(1)}$ and $f^{(2)}$, respectively. Then there exists an instance (r, s) (over R), having size $|r| = |r^{(1)}| + |r^{(2)}|$ and $|s| = |s^{(1)}| + |s^{(2)}|$, with indicator function $f_{(r,s)} = f^{(1)} \wedge f^{(2)}$.*

Proof. W.l.o.g. assume that the values appearing in $r^{(1)}$ and $s^{(1)}$ are disjoint from those in $r^{(2)}$ and $s^{(2)}$. We straightforwardly construct instance (r, s) by defining $r = r^{(1)} \cup r^{(2)}$ and $s = s^{(1)} \cup s^{(2)}$. Obviously, the construction matches the requirements on both the computability and size of the resulting instance. It remains to show that $f_{(r,s)} = f^{(1)} \wedge f^{(2)}$.

Equivalently, we show that X is an inclusion dependency in (r, s) if and only if it is one in both sub-instances $(r^{(1)}, s^{(1)})$ and $(r^{(2)}, s^{(2)})$. First, suppose the condition holds. Then, for every tuple $r_i^{(1)} \in r^{(1)}$, there exists a tuple $s_j^{(1)} \in s^{(1)}$ with $r_i^{(1)}[X] = s_j^{(1)}[X]$, and the same holds for $r^{(2)}$ and $s^{(2)}$. As all said tuples are also present in (r, s), X is an inclusion dependency there as well. Conversely, suppose X is not an inclusion dependency in, say, $(r^{(1)}, s^{(1)})$. Then $r^{(1)}$ has a tuple $r_i^{(1)}$ such that $r_i^{(1)}$ disagrees on X with *every* $s_j^{(1)} \in s^{(1)}$. By construction, $r_i^{(1)}$ is also in r. Moreover, all tuples in s belong either to $s^{(1)}$ or have completely disjoint values. This results in $r_i^{(1)}[X] \neq s_j[X]$ for every $s_j \in s$ as desired. ◀

One could hope that there is a similar method treating disjunctions. However, we believe that there is none that is both computable in FPT-time and compatible with a complementing method for conjunctions (e.g. the one shown above). The reasoning is as follows: Negative literals are easily expressible as instances of $\text{IND}_{\textit{fixed}}$ using pairs of single-tuple relations. Together with FPT-time procedures of constructing conjunctions as well as disjunctions one could encode antimonotone Boolean formulas of *arbitrary logical depth.* According to the Antimonotone Collaps Theorem by Downey and Fellows [11] this would render $\text{IND}_{\textit{fixed}}$ to be hard for *all* classes $W[t]$. As a consequence of Theorem 5 (in conjunction with Lemma 4) the W-hierarchy would collapse to the level $W[3]$. That being said, there is a method specifically tailored to antimonotone formulas in DNF.

▶ **Lemma 8.** *Let φ be an antimonotone formula in DNF. Then there is an instance (r, s) for problem* $\text{IND}_{\textit{fixed}}$ *of size polynomial in $|\varphi|$ such that $f_\varphi = f_{(r,s)}$.*

Proof. Let $x_1, \dots, x_n$ be the variables of formula φ. Define schema $R = \{A_1, \dots, A_n\}$ by identifying each variable x_i with attribute A_i. To build the instance (r, s) over the schema R, we first describe the relation r and then construct s accordingly.

As φ is a DNF formula, it is the disjunction of conjunctive clauses $c_1, \dots, c_m$. For each clause c_j, we create the tuple r_j by setting $r_j[A_i] = j$ if x_i occurs in c_j and $r_j[A_i] = 0$ otherwise. For example, the clause $c_1 = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$ in Figure 3 leads to the tuple $(1, 1, 1, 0, 0, 0)$. Relation s is obtained by first creating m copies of r. In the j-th copy we then set the value for attribute A_i to the special symbol "$-$" whenever x_i occurs in the conjunctive clause c_j; see Figure 3 again. Note that $|R|$ equals the number of variables of φ and $|r|$ is linear while $|s|$ is quadratic in the number of conjunctive clauses in φ. In total, the size of the instance (r, s) is polynomial in $|\varphi|$. It is left to show that $f_\varphi = f_{(r,s)}$.

First, suppose $f_\varphi(X) = 1$ for some binary vector X of length $|R|$ or, equivalently, a subset $X \subseteq R$. We show that $f_{(r,s)}(X) = 1$. Necessarily, we have $f_{c_j}(X) = 1$ for at least one conjunctive clause c_j and since the clause contains only negative literals, all of its variables evaluate to 0. This is equivalent to X not containing any attributes corresponding to variables occurring in c_j. In the j-th copy of r in s the values were changed to "$-$" for exactly those attributes. Thus, restriction $s[X]$ comprises an exact copy of $r[X]$, resulting in $f_{(r,s)}(X) = 1$ by definition.

$$\varphi = c_1 \lor c_2 \lor c_3$$
$$c_1 = (\neg x_1 \land \neg x_2 \land \neg x_3)$$
$$c_2 = (\neg x_2 \land \neg x_4 \land \neg x_5)$$
$$c_3 = (\neg x_1 \land \neg x_3 \land \neg x_4 \land \neg x_6)$$

A_1	A_2	A_3	A_4	A_5	A_6
1	1	1	0	0	0
0	2	0	2	2	0
3	0	3	3	0	3

A_1	A_2	A_3	A_4	A_5	A_6
−	−	−	0	0	0
−	−	−	2	2	0
−	−	−	3	0	3
1	−	1	−	−	0
0	−	0	−	−	0
3	−	3	−	−	3
−	1	−	−	0	−
−	2	−	−	2	−
−	0	−	−	0	−

Figure 3 Illustration of Lemma 8. Formula φ is on the left (with the three conjunctive clauses c_1, c_2, c_3), relation r in the center and relation s on the right.

For the opposite direction, suppose $f_\varphi(X) = 0$. Then, for each conjunctive clause, at least one variable evaluates to 1 and, consequently, for each tuple in s, the value of at least one attribute in X was replaced by "−". As r does not contain the special symbol "−" at all, X is not an inclusion dependency, i.e., $f_{(r,s)}(X) = 0$. ◀

Lemma 8 in combination with Lemma 7 implies that, given an antimonotone 3-normalized formula φ, we can build an instance (r, s) of problem $\mathrm{IND}_{\mathit{fixed}}$ in FPT-time (even polynomial) such that $f_\varphi = f_{(r,s)}$. Using the findings of Section 4.1, this proves the desired theorem.

▶ **Theorem 9.** $\mathrm{IND}_{\mathit{fixed}}$ *and* IND *are* $W[3]$*-complete.*

5 Conclusion

We have determined the complexity of various dependency problems when parameterized by the solution size. Our results imply that these problems do *not* admit FPT algorithms unless the W-hierarchy at least partially collapses. This is unfortunate, the choice of parameter appears to be very natural in the sense that the requirement of a small solution size is regularly met in practice (Figure 1). Notwithstanding our results, one can still obtain FPT algorithms by using *other* parameters. As an example, to solve the problem UNIQUE for a relation r over the schema R (and similar considerations hold for FD and IND), one can consider all subsets of R and check for each whether it is a unique column combination. This takes polynomial time for each of the $2^{|R|}$ subsets. Thus, this leads to an FPT-algorithm with $|R|$ as parameter. This is of course not very satisfying, as assuming $|R|$ to be small is a much stronger assumption than assuming the solution size to be small.

Similarly, one could consider the maximum number d of attributes on which two tuples in a relation r disagree. Then any pair of tuples yields up to d candidate attributes such that at least one of these attributes must be contained in every unique column combination. Thus, one can check whether there is a solution of size k to the problem UNIQUE by using a bounded search tree of height k with nodes of degree at most d. This gives an FPT algorithm with respect to the parameter $d + k$. However, assuming that any two pairs in a relation differ only on a few columns seems to be an unrealistic assumption for most data sets.

We leave it as an open problem for future research to figure out properties of realistic instances that can explain and hopefully even improve the running times of practical methods for dependency detection in relational databases. For example, by designing a multivariate algorithm with more than one parameter.

Acknowledgement. We would like to thank Sebastian Kruse, Felix Naumann and Thorsten Papenbrock for the interesting discussions that initiated our research on this topic, for their continued support and of course for providing us with the data presented in the introduction. We also want to thank Christoph Keßler for proof-reading an early draft of this paper and Sharon Nemeth for revising the final version.

References

1 Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: a survey. *The VLDB Journal*, 24:557–581, 2015.

2 Tatsuya Akutsu and Feng Bao. Approximating minimum keys and optimal substructure screens. In *Proceedings of the 2nd Annual International Conference on Computing and Combinatorics (COCOON)*, volume 1090 of *LNCS*, pages 290–299. Springer, 1996. doi:10.1007/3-540-61332-3_163.

3 Catriel Beeri, Martin Dowd, Ronald Fagin, and Richard Statman. On the structure of armstrong relations for functional dependencies. *Journal of the ACM*, 31:30–46, 1984. doi:10.1145/2422.322414.

4 Jianer Chen and Fenghui Zhang. On product covering in 3-tier supply chain models: Natural complete problems for $W[3]$ and $W[4]$. *Theoretical Computer Science*, 363:278–288, 2006. doi:10.1016/j.tcs.2006.07.016.

5 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 C. J. Date. *An Introduction to Database Systems*. Addison-Wesley Longman Publishing, Boston, MA, USA, 8th edition, 2003.

7 Scott Davies and Stuart Russell. NP-completeness of searches for smallest possible feature sets. Technical report, AAAI, 1994.

8 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–178, 1992.

9 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24:873–921, 1995.

10 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theoretical Computer Science*, 141:109–131, 1995.

11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

12 Thomas Eiter and Georg Gottlob. Hypergraph transversal computation and related problems in logic and AI. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence*, pages 549–564, 2002. doi:10.1007/3-540-45757-7_53.

13 Martin Grohe. The parameterized complexity of database queries. In *Proceedings of the 20th Symposium on Principles of Database Systems*, pages 82–92, 2001. doi:10.1145/375551.375564.

14 Martti Kantola, Heikki Mannila, Kari-Jouko Räihä, and Harri Siirtola. Discovering functional and inclusion dependencies in relational databases. *International Journal of Intelligent Systems*, 7:591–607, 1992. doi:10.1002/int.4550070703.

15 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972. URL: http://www.cs.berkeley.edu/~luca/cs172/karp.pdf.

16 David Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

17 Christos H. Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58:407–427, 1999. doi:10.1006/jcss.1999.1626.

18 Aaron Swartz. MusicBrainz: a semantic web service. *IEEE Intelligent Systems*, 17:76–77, 2002. See www.musicbrainz.org. doi:10.1109/5254.988466.