The Complexity of Dependency Detection and Discovery in Relational Databases*

Thomas Bläsius[a,1], Tobias Friedrich[b,1], Martin Schirneck[b,1,*]

[a]*Karlsruhe Institute of Technology, Karlsruhe, Germany*
[b]*Hasso Plattner Institute, University of Potsdam, Potsdam, Germany*

Abstract

Multi-column dependencies in relational databases come associated with two different computational tasks. The detection problem is to decide whether a dependency of a certain type and size holds in a given database, the discovery problem asks to enumerate all valid dependencies of that type. We settle the complexity of both of these problems for unique column combinations (UCCs), functional dependencies (FDs), and inclusion dependencies (INDs).

We show that the detection of UCCs and FDs is W[2]-complete when parameterized by the solution size. The discovery of inclusion-wise minimal UCCs is proven to be equivalent under parsimonious reductions to the transversal hypergraph problem of enumerating the minimal hitting sets of a hypergraph. The discovery of FDs is equivalent to the simultaneous enumeration of the hitting sets of multiple input hypergraphs.

We further identify the detection of INDs as one of the first natural W[3]-complete problems. The discovery of maximal INDs is shown to be equivalent to enumerating the maximal satisfying assignments of antimonotone, 3-normalized Boolean formulas.

Keywords: data profiling, enumeration complexity, functional dependency,

---

inclusion dependency, parameterized complexity, parsimonious reduction, transversal hypergraph, unique column combination, W[3]-completeness

---

1. Introduction

Data profiling is the extraction of metadata from databases. An important class of metadata in the relational model are multi-column dependencies. They describe interconnections between the values stored for different attributes or even across multiple instances. Arguably the most prominent type of such dependencies are the unique column combinations (UCCs), also known as candidate keys. These are collections of attributes such that the value combinations appearing in those attributes identify all rows of the database. In Figure 1, the `Name` and `Area Code` provide a unique fingerprint of the first database as, for example, there is only one `Doe,John` living in `UK-W1K`. Note that none of the two columns can be left out due to duplicate values. Small UCCs are natural candidates for primary keys, avoiding the need to introduce surrogate identification. More importantly though, knowledge of the inclusion-wise minimal unique column combinations enable various data cleaning tasks as well as query optimization. For example, value combinations of UCCs are by definition distinct and form groups of size 1, thus SQL queries working on UCCs can skip the grouping phase and the `DISTINCT` operation, even if requested by the user [12, 47]. Also, the presence of UCCs allows for early returns of `SELECT` and `ORDER BY` operations.

Unfortunately, a dataset only rarely comes annotated with its dependencies. Much more often they need to be computed from raw data. This leads to two different computational tasks. The *detection problem*, is to decide for a given database whether it admits a UCC with only a few columns. The *discovery problem* instead asks for a complete list of all minimal UCCs, regardless of their size and number. An equivalent term for the latter, which is probably more common in the algorithms community, is the *enumeration* of minimal solutions.

Functional dependencies (FDs), another dependency type, model the case in which one is only interested in identifying the values of a specific column, instead of all columns. In the example in Figure 1, it is enough to know the `Area Code` to also infer the `City` since the value in the former determines the latter. Finally, inclusion dependencies (INDs) reveal connections between different databases. A unary IND holds if all values in a column of

2

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	W Broadway 400	San Diego	US-CA-92101
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

ID	Author	Title
25	Doe, John	The Art of Computer Programming
33	Menigmand, Morten	Prinsessen paa Ærten
47	Nightingale, Florence	Cassandra
76	Mustermann, Max	Grundzüge der Theoretischen Logik
90	Lovelace, Ada K.	Sketch of the Analytical Engine

Figure 1: Illustration of multi-column dependencies. `Name` and `Area Code` together are a minimal unique column combination in the first database. The functional dependency `Name, Area Code → City` holds, but its left-hand side is not minimal since `Area Code → City` is also valid. There are two maximal inclusion dependencies of size 1 between the first and the second database, `Age` is included in `ID` and `Name` in `Author`. They cannot be combined to an inclusion dependency of size 2 since, e.g., the value combination (33, `Doe, John`) does not appear in column combination `ID, Author`.

the first database are also contained in one column of the second database. In Figure 1, all values of the `Name` attribute appear again for the `Author` attribute in the second table. The IND has higher arity if the inclusion also pertains to the tuples of values in multiple columns.[1] In contrast to UCCs and FDs, where we want solutions to be small, we ought to find large or maximal inclusion dependencies. Those are much more likely to be caused by the inherent structure of the data than by mere coincidence.

Similar to UCCs, discovering the functional dependencies and inclusion dependencies of a database (respectively pairs thereof) is an important pre-processing step intended to improve the subsequent data access. The knowledge of the valid FDs and INDs is used for example in cardinality estimation in query plan optimizers [35], query rewriting [33], and joins [10]. For a detailed exposition of the applications of data dependencies, we refer the reader to the recent survey by Kossmann, Papenbrock, and Naumann [41].

The detection (decision) problems for all three types of dependencies are

---

[1]The formal definitions of UCCs, FDs, and INDs can be found in Section 2.2.

NP-complete.[2] Notwithstanding, detection algorithms often perform well on practical datasets [1]. One approach to bridge this apparent gap is to analyze whether properties that are usually observed in realistic data benevolently influence the hardness of the problem. Exploiting those properties may even lead to algorithms that guarantee a polynomial running time in case these features are present in the problem instance. This is formalized in the concept of parameterized algorithms [18, 25, 45]. There, one tries to ascribe the exponential complexity entirely to an parameter of the input, other than its mere size. If the parameter is bounded for a given class of instances, we obtain a polynomial running time whose order of growth is independent of the parameter. This, of course, requires one to identify parameters that are both algorithmically exploitable and small in practice.

Consider, for instance, the histograms in Figure 2, showing the size distribution of minimal unique column combinations and functional dependencies, as well as maximal inclusion dependencies in the MusicBrainz database [52]. The majority of functional dependencies are rather small, same for unique column combinations and inclusion dependencies. Beside surrogate keys, giving rise to multiple functional dependencies of size 1, causalities in the data can also lead to small FDs. For example, the name of an event together with the year in which it started determines the year in which it ends, implying an FD of size 2. Note that the starting year alone is usually not enough to infer this information. The name of the action, however, seems to indicate whether the event ends in the same year or the next. The size of the dependency is thus a natural candidate for an algorithmic parameter. Notwithstanding, we show that it is unlikely to be the sole explanation for the good practical performance. We prove that the detection of unique column combinations and functional dependencies is $\mathsf{W}[2]$-complete with respect to the size of the sought solution, detecting inclusion dependencies is even $\mathsf{W}[3]$-complete. For all we know, this excludes any algorithm parameterized by the size.

The hardness of detecting INDs is surprising also from a complexity-theoretic standpoint. Currently, there are only a handful of natural problems known to be complete for the class $\mathsf{W}[3]$. The first one was given by Chen and Zhang [14] in the context of supply chain management. We show here that the detection of inclusion dependencies has this property, making it the second such problem. Since this result was first announced, Bläsius et al. [6] have also

---

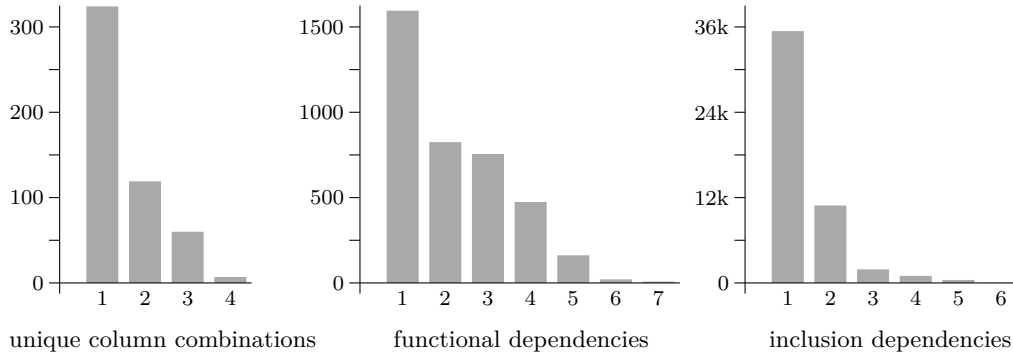[2]See Sections 3.1 and 4.1 for precise statements and references.

4

Figure 2: The number of minimal unique column combinations, minimal functional dependencies, and maximal inclusion dependencies for varying solution sizes in the MusicBrainz database.

proven the extension problem for minimal hitting sets to be $\mathsf{W}[3]$-complete using different techniques. The latter has subsequently been improved by Casel et al. [11], they have shown that already the special case of extension to minimal dominating sets in bipartite graphs is hard for $\mathsf{W}[3]$. Finally, building on the work presented here, Hannula, Song, and Link [34] have very recently identified independence detection in relational databases as another representative of this class.

For enumeration problems, the border of tractability does not run between polynomial and super-polynomial time, at least not when measured in the input only. The number of solutions one wishes to be computed may be exponential in the input size, ruling out any polynomial algorithm. Instead, one could ask for an algorithm that scales polynomially both in the input and the number of solutions. The most important yardstick in enumeration complexity is the *transversal hypergraph* problem, where one is tasked to compute all minimal hitting sets of a given hypergraph. Currently, the fastest known algorithm runs in time $N^{\mathrm{O}(\log N/\log\log N)}$, where N is the combined input and output size [30]. It is the major open question in the field of enumeration algorithms whether the transversal hypergraph problem can be solved in output-polynomial time. Besides data profiling, the problem also emerges in many other applications in artificial intelligence [50], machine learning, distributed systems, and monotone logic [28], as well as bioinformatics [49]. There are many algorithms known that work well on practical instances [32].

We use the insights gained on the detection of multi-column dependencies in databases to also investigate their discovery (enumeration) problems. It is

known that minimal unique column combinations and functional dependencies can be discovered in output-polynomial time if and only if the transversal hypergraph problem has an output-polynomial solution [26]. However, this was proven via a Turing-style reduction that continuously calls a decision subroutine to check whether the enumeration part has already found all solutions. This construction inherently uses space proportional to the output and is therefore hardly useful in practice. We are able to radically simplify this equivalence and connect unique column combinations, functional dependencies, and hitting sets directly at the enumeration level using so-called *parsimonious reductions*, running in polynomial time and space. We give similar results also for the discovery of maximal inclusion dependencies.

Parsimonious reductions are the most restrictive form of reduction between enumeration problems, but–therefore–also the most useful in practice. The close connection to the transversal hypergraph problem that we prove in this work explains in parts why dependency discovery works quite well on real-world databases. Moreover, it allows us to transfer ideas from the design of hitting set enumeration algorithms to data profiling, thereby connecting the two research areas. For example, there are very space-efficient algorithms known for the transversal hypergraph problem, while memory consumption still seems to be a major obstacle in dependency discovery [46, 53]. We hope that our results here inspire better data profiling algorithms in the future.

Our Contribution. We settle the parameterized complexity of the cardinality-constrained decision problems for unique column combinations, functional dependencies, and inclusion dependencies in relational databases, with the solution size as parameter. We prove the following theorems.

Theorem 1. *Detecting a unique column combination of size k in a relational database is $\mathsf{W}[2]$-complete when parameterized by k. The same is true for the detection of a valid, non-trivial functional dependency with a left-hand side of size at most k, even if the desired right-hand side is given in the input.*

Theorem 2. *Detecting an inclusion dependency of size k in a pair of relational databases is $\mathsf{W}[3]$-complete when parameterized by k. The result remains true even if both databases are over the same relational schema with the identity mapping between their columns.*

We also characterize the complexity of enumerating all multi-column dependencies of a certain type in a given database. We do so by proving

6

parsimonious equivalences with well-known enumeration problems as well as generalizations thereof.

Theorem 3. *The following enumeration problems are equivalent under parsimonious reductions:*

   (i) discovering all minimal unique column combinations of a relational database;

  (ii) discovering all minimal, valid, and non-trivial functional dependencies with a fixed right-hand side;

 (iii) the transversal hypergraph problem.

The discovery of functional dependencies with arbitrary right-hand sides is equivalent to enumerating the hitting sets of multiple input hypergraphs. The latter two problems are at least as hard as the transversal hypergraph problem.

Theorem 4. *The following enumeration problems are equivalent under parsimonious reductions:*

  (i) discovering all maximal inclusion dependencies of a pair of relational databases;

 (ii) enumerating all maximal satisfying assignments of an antimonotone, 3-normalized Boolean formula.

This remains true even if the two databases are over the same schema and only inclusions between the same columns are allowed. All those problems are at least as hard as the transversal hypergraph problem.

Finally, we also briefly discuss the consequences of our findings to the approximability of minimum dependencies.

Outline. In the next section, we fix some notation and review basic concepts needed for the later proofs. Section 3 treats unique column combinations and functional dependencies, Section 4 then considers inclusion dependencies. Both of these sections start with a segment that formally defines all decision and enumeration problems discussed in the respective section. The paper is concluded in Section 5.

7

2. Preliminaries

2.1. Hypergraphs and Hitting Sets

A *hypergraph* is a non-empty, finite *vertex set* $V \neq \emptyset$ together with a set of subsets $\mathcal{H} \subseteq \mathcal{P}(V)$, the *(hyper-)edges*. We identify a hypergraph with its edge set $\mathcal{H}$ if this does not create any ambiguities. We do not exclude special cases of this definition like the empty graph ($\mathcal{H} = \emptyset$), an empty edge ($\emptyset \in \mathcal{H}$), or isolated vertices ($V \supsetneq \bigcup_{E \in \mathcal{H}} E$). A hypergraph is *Sperner* if none of its edges is contained in another. The *minimization* of $\mathcal{H}$ is the subset of all inclusion-wise minimal edges, $\min(\mathcal{H}) = \{E \in \mathcal{H} \mid \forall E' \in \mathcal{H} \colon E' \subseteq E \Rightarrow E' = E\}$. This should not be confused with the notation for a minimum element of a set. The minimization of a hypergraph is always Sperner.

A *hitting set*, or *transversal*, of a hypergraph $(V, \mathcal{H})$ is a set $T \subseteq V$ of vertices such that T has a non-empty intersection with every edge $E \in \mathcal{H}$. A hitting set is *(inclusion-wise) minimal* if it does not properly contain another hitting set. The minimal hitting sets of $\mathcal{H}$ form a Sperner hypergraph on the same vertex set V, the *transversal hypergraph* $\mathrm{Tr}(\mathcal{H})$. Regarding transversals, it does not make a difference whether the full hypergraph $\mathcal{H}$ is considered or its minimization, it holds that $\mathrm{Tr}(\min(\mathcal{H})) = \mathrm{Tr}(\mathcal{H})$.

2.2. Relational Databases and Dependencies

A *(relational) schema* R is a non-empty, finite set of *attributes* or *columns*. Each attribute comes implicitly associated with a set of admissible values. A *row*, or *record*, over schema R is a tuple r whose entries are indexed by R such that, for any attribute $a \in R$, the value $r[a]$ is admissible for a. A *(relational) database* $\mathfrak{r}$ over R is a finite set of records. For any row r and subset $X \subseteq R$ of columns, $r[X]$ is the subtuple of r projected onto S. We let $\mathfrak{r}[X]$ denote the family of all such subtuples of rows in $\mathfrak{r}$. Note that $\mathfrak{r}[X]$ is a multiset as the same combination of values may appear in different rows.

We are interested in multi-column dependencies in relational databases, namely, unique column combinations, functional dependencies, and inclusion dependencies. For their definition, we need the notion of difference sets. For any two distinct rows r_1, r_2, $r_1 \neq r_2$, over the same schema R, their *difference set* $\{a \in R \mid r_1[a] \neq r_2[a]\}$ is the set of attributes in which the rows disagree. A difference set is *(inclusion-wise) minimal* if it does not properly contain a difference set for another pair of rows in the database. We denote the hypergraph of minimal difference sets by $\mathcal{D}$. The vertex set is R.

A *unique column combination* (UCC), or simply *unique*, for some database $\mathfrak{r}$ over schema R, is a subset $U \subseteq R$ of attributes such that for any two records $r, s \in \mathfrak{r}$, $r \neq s$, there is an attribute $a \in U$ such that $r[a] \neq s[a]$. Equivalently, U is a UCC for $\mathfrak{r}$ iff the collection $\mathfrak{r}[U]$ of projections onto U is a mere set, that is, every tuple appears at most once. A UCC is *(inclusion-wise) minimal* if it does not properly contain another UCC. There is an intimate connection between UCCs and transversals in hypergraphs. It is well-known in the literature, see [1, 19], probably the first explicit mention was by Mannila and Räihä [43].

Observation 5 (Folklore). *The unique column combinations are the hitting sets of difference sets. In particular, let $\mathfrak{r}$ be a database and $\mathcal{D}$ the hypergraph of its minimal difference sets. Then, any minimal transversal in $\mathrm{Tr}(\mathcal{D})$ is a minimal unique of $\mathfrak{r}$ and there are no other minimal uniques in $\mathfrak{r}$.*

Functional dependencies (FDs) over a schema R are expressions of the form $X \to a$ for some set $X \subseteq R$ of columns and a single attribute $a \in R$. The set X is the *left-hand side* (LHS) of the dependency and a the *right-hand side* (RHS). We say that the FD has *size $|X|$*. A functional dependency $X \to a$ is said to *hold*, or be *valid*, in an database $\mathfrak{r}$ (over R) if any pair of records that agree on X also agree on a, that is, if $r[X] = s[X]$ implies $r[a] = s[a]$ for any $r, s \in \mathfrak{r}$. Otherwise, $X \to a$ is said to *fail* in $\mathfrak{r}$, or be *invalid*. The FD $\emptyset \to a$ holds iff all rows agree on a. An FD $X \to a$ is *(inclusion-wise) minimal* if it holds in $\mathfrak{r}$ and $X'' \to a$ fails for any proper subset $X'' \subsetneq X$. A functional dependency is *non-trivial* if $a \notin X$. Note that trivial functional dependencies hold in any database.

Finally, we turn to multi-column dependencies among multiple databases. Let R and S be two relational schemas and $\mathfrak{r}$ and $\mathfrak{s}$ databases over R and S, respectively. For some $X \subseteq R$, let $\sigma \colon X \to S$ be an injective map. The pair (X, σ) is an *inclusion dependency* (IND) if, for each row $r \in \mathfrak{r}$, there exists some $s \in \mathfrak{s}$ such that $r[a] = s[\sigma(a)]$ for every $a \in X$, that is, iff the inclusion $\mathfrak{r}[X] \subseteq \mathfrak{s}[\sigma(X)]$ holds. If the map σ is given in the input, we say that X is the dependency. Such an inclusion dependency then is *maximal* if the set X is maximal among all INDs for $\mathfrak{r}$ and $\mathfrak{s}$. For the general case, we define a partial order on the pairs (X, σ). We say that $(X, \sigma) \preccurlyeq (X', \sigma')$ holds if $X \subseteq X'$ and σ is the restriction of σ' to X. An inclusion dependency then is *maximal* if it is an $\preccurlyeq$-maximal element among the inclusion dependencies between $\mathfrak{r}$ and $\mathfrak{s}$. Observe that the inclusion dependencies are indeed downward closed

with respect to $\preccurlyeq$. However, it may happen that (X, σ) and (X', σ') are both maximal although $X' \subsetneq X$ is a strict subset.

2.3. Parameterized Complexity

The central idea of the *parameterized complexity* of a decision problem is to identify a quantity of the input, other than its mere size, that captures the hardness of the problem. The decision problem associated with a language $\Pi \subseteq \{0,1\}^*$ is to determine whether some instance $I \in \{0,1\}^*$ is in Π. A decision problem is *parameterized* if any instance I is additionally augmented with a *parameter* $k = k(I) \in \mathbb{N}^+$, we thus have $\Pi \subseteq \{0,1\}^* \times \mathbb{N}^+$. A parameterized decision problem Π is *fixed-parameter tractable* (FPT), if there exists a computable function $f \colon \mathbb{N}^+ \to \mathbb{N}^+$ and an algorithm that decides any instance (I, k) in time $f(k) \cdot \mathsf{poly}(|I|)$. The complexity class FPT is the collection of all fixed-parameter tractable problems.

Let Π and Π' be two parameterized problems. A *parameterized reduction*, or *FPT-reduction*, from Π to Π' is an algorithm running in time $f(k) \cdot \mathsf{poly}(|I|)$ on instances (I, k), which outputs some instance (I', k') such that $k' \leq g(k)$ for some computable function $g \colon \mathbb{N}^+ \to \mathbb{N}^+$, and $(I, k) \in \Pi$ holds if and only if $(I', k') \in \Pi'$ does. Due to the time bound, we have $|I'| \leq f(k)\,\mathsf{poly}(|I|)$. If there is also an FPT-reduction from Π' to Π, we say that the problems are *FPT-equivalent*. A notable special case of FPT-reductions are polynomial many-one reductions that preserve the parameter, meaning $k' = k$.

Parameterized reductions give rise to the so-called W-*hierarchy* of complexity classes. We use one of several equivalent definitions involving (mainly) Boolean formulas. However, first consider the INDEPENDENT SET problem on graphs parameterized by the size of the sought solution. The class $\mathsf{W}[1]$ is the collection of all parameterized problems that admit a parameterized reduction to INDEPENDENT SET. For some positive integer t, a Boolean formula is *t-normalized* if it is a conjunction of disjunctions of conjunctions of disjunctions (and so on) of literals with $t-1$ alternations or, equivalently, t levels in total. The WEIGHTED t-NORMALIZED SATISFIABILITY problem is to decide, for a t-normalized formula φ and a positive integer k, whether φ admits a satisfying assignment of Hamming weight k, that is, with (exactly) k variables set to TRUE. Here, k is the parameter. For every $t \geq 2$, $\mathsf{W}[t]$ is the class of all problems reducible to WEIGHTED t-NORMALIZED SATISFIABILITY. The classes $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots$ form an ascending hierarchy. All inclusion are conjectured to be strict, which is however still unproven.

Chen et al. [13] showed that, assuming the Exponential Time Hypothesis[3] (ETH), INDEPENDENT SET with budget k on n-vertex, m-edge graphs cannot be solved in time $f(k) \cdot (m+n)^{o(k)}$. Hence, ETH implies $\mathsf{FPT} \neq \mathsf{W}[1]$.

2.4. Enumeration Complexity

Enumeration is the task of compiling and outputting a list of all solutions to a computational problem without repetitions. Note that this is different from a counting problem, which asks for the mere number of solutions. More formally, an *enumeration problem* is a function $\Pi \colon \{0,1\}^* \to \mathcal{P}(\{0,1\}^*)$ such that, for all instances $I \in \{0,1\}^*$, the set of *solutions* $\Pi(I)$ is finite. An algorithm solving this problem needs to output, on input I, all elements of $\Pi(I)$ exactly once. We do not impose any order on the output. We focus on the enumeration of minimal hitting sets, that is, $\Pi \colon (V, \mathcal{H}) \mapsto \mathrm{Tr}(\mathcal{H})$.

An *output-polynomial* algorithm runs in time polynomial in the combined input and output size $N = |V| + |\mathcal{H}| + |\mathrm{Tr}(\mathcal{H})|$. A (seemingly) stronger requirement is an *incremental polynomial* algorithm, generating the solutions in such a way that the i-th *delay*, the time between the $(i-1)$-st and i-th output, is bounded by $\mathsf{poly}(|V|, |\mathcal{H}|, i)$. This includes the *preprocessing time* until the first solution arrives ($i = 1$) as well as the *postprocessing time* between the last solution and termination ($i = |\mathrm{Tr}(\mathcal{H})| + 1$). The strongest form of output-efficiency is that of *polynomial delay*, where the delay is universally bounded by $\mathsf{poly}(|V|, |\mathcal{H}|)$. There is currently no output-polynomial algorithm known for the transversal hypergraph problem, but its existence would immediately imply also an incremental polynomial algorithm [4].

Arguably the most restrictive way to relate enumeration problems are parsimonious reductions. The concept is closely related but should not be confused with the homonymous class of reductions for counting problems [9]. A *parsimonious reduction* from enumeration problem Π to Π' is a pair of polynomial time computable functions $f \colon \{0,1\}^* \to \{0,1\}^*$ and $g \colon (\{0,1\}^*)^2 \to \{0,1\}^*$ such that, for any instance $I \in \{0,1\}^*$, $g(I, \cdot)$ is a bijection from $\Pi'(f(I))$ to $\Pi(I)$. The behavior of $g(I, \cdot)$ on $\{0,1\}^* \backslash \Pi'(f(I))$ is irrelevant. Intuitively, any enumeration algorithm for Π' can then be turned into one for Π by first mapping the input I to $f(I)$ and translating the solutions back via g. Observation 5 establishes a parsimonious reduction from

---

[3]The Exponential Time Hypothesis is the conjecture that the satisfiability problem of formulas in conjunctive normal form on n variables with 3 literals per clause (3-SAT) cannot be solved in time $2^{o(n)}$ [36, 37].

the enumeration of minimal UCCs to the transversal hypergraph problem with $f\colon (R, \mathfrak{r}) \mapsto (R, \mathcal{D})$ and $g((R, \mathfrak{r}), \cdot)$ being the identity over subsets of R.

3. Unique Column Combinations and Functional Dependencies

Theoreticians as well as practitioners in data profiling and database design are frequently confronted with the task of finding a small collection of items that has a non-empty intersection with each member of a prescribed family of sets, see [1, 19, 20, 38, 42]. They thus aim to solve instances of the hitting set problem. In this section, we show that this encounter is inevitable in the sense that detecting a small unique column combination or functional dependency in a relational database is the same as finding a hitting set in a hypergraph. Even more so, this equivalence extends to enumeration. We show that the associated discovery problems of finding all UCCs or FDs is indeed the same as enumerating all hitting sets.

We first formally define the respective decision and enumeration problems. The decision versions are always parameterized by the solution size. We then order them in a (seemingly) ascending chain via parameterized reductions. However, the start and end points of this chain will turn out to be FPT-equivalent, settling the complexity of the problems involved as complete for the parameterized complexity class $\mathsf{W}[2]$. We then also show the equivalence of the corresponding enumeration problems under parsimonious reductions.

3.1. Problem Definitions

Recall the definitions of hitting sets as well as unique column combinations and functional dependencies from Sections 2.1 and 2.2. We are interested in the parameterized complexity of the associated cardinality-constrained decision problems. The constraint always serves as the parameter.

HITTING SET

  Instance: A hypergraph $(V, \mathcal{H})$ and a non-negative integer k.

Parameter: The non-negative integer k.

  Decision: Is there a set $T \subseteq V$ of vertices with cardinality $|T| = k$
             such that T is a hitting set for $\mathcal{H}$?

Note that if $k > |V|$, then the answer to the decision problem is trivially FALSE; otherwise, there is no difference between deciding the existence of a

transversal with *at most* or *exactly* k elements since every superset of a hitting set is again a hitting set. We ignore the special case of a too large k since parameterized complexity is primarily concerned with the situation that the parameter is much smaller than the input size. The unparameterized Hitting Set problem is one of Karp's initial 21 NP-complete problems [39]. In fact, its minimization variant–to compute the minimum size of any hitting set–is even NP-hard to approximate within a factor of $(1 - \varepsilon) \ln |V|$ for any constant $\varepsilon > 0$ [21]. The parameterized variant defined above is the prototypical W[2]-complete problem [25].

The corresponding enumeration problem broadens the notion of a "small" solution, namely, the task is now to list all *inclusion-wise minimal* hitting sets, that is, the edges of the transversal hypergraph $\mathrm{Tr}(\mathcal{H})$. All other (non-minimal) hitting sets can be trivially obtained from the minimal ones by arbitrarily adding more vertices. Note that the term *enumeration* implicitly contains the requirement not to repeat outputs.

Transversal Hypergraph

Instance: A hypergraph $(V, \mathcal{H})$.

Enumeration: List all edges of $\mathrm{Tr}(\mathcal{H})$.

Let $N = |\mathcal{H}| + |\mathrm{Tr}(\mathcal{H})| + |V|$ denote the combined input and output size, Fredman and Kachiyan's algorithm solves Transversal Hypergraph in time $N^{\mathrm{O}(\log N / \log \log N)}$ [30].

We generalize the problem to the enumeration of minimal hitting sets for multiple input hypergraphs simultaneously. We do not prescribe any order in which the hypergraphs are processed. However, we want to be able to quickly tell to which input a solution belongs. For this, let $\mathrm{Tr}(\mathcal{H}) \mathbin{\dot\cup} \mathrm{Tr}(\mathcal{G})$ denote the disjoint union of the transversal hypergraphs of $(V, \mathcal{H})$ and $(W, \mathcal{G})$ with the additional requirement that the union is encoded in such a way that, for any $T \in \mathrm{Tr}(\mathcal{H}) \mathbin{\dot\cup} \mathrm{Tr}(\mathcal{G})$, the containment $T \in \mathrm{Tr}(\mathcal{H})$ is decidable in time $\mathsf{poly}(|V|, |W|, |\mathcal{H}|, |\mathcal{G}|)$, independently of the sizes $|\mathrm{Tr}(\mathcal{H})|$ and $|\mathrm{Tr}(\mathcal{G})|$.

Transversal Hypergraph Union

Instance: A d-tuple of hypergraphs $(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_d)$.

Enumeration: List all edges of $\mathrm{Tr}(\mathcal{H}_1) \mathbin{\dot\cup} \mathrm{Tr}(\mathcal{H}_2) \mathbin{\dot\cup} \ldots \mathbin{\dot\cup} \mathrm{Tr}(\mathcal{H}_d)$.

We now define the detection and discovery problems of multi-column dependencies in relational databases, again starting with the cardinality-

constraint decision problems.

UNIQUE COLUMN COMBINATION

Instance: A relational database $\mathfrak{r}$ over schema R
and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $U \subseteq R$ of attributes with cardinality $|U| = k$
such that U is a unique column combination in $\mathfrak{r}$?

The minimization variant of UNIQUE COLUMN COMBINATION is NP-hard [51].

ENUMERATE MINIMAL UCCS

Instance: A relational database $\mathfrak{r}$.

Enumeration: List all minimal unique column combinations of $\mathfrak{r}$.

For functional dependencies, we define two variants of the decision problem that slightly differ in the given input. The first one fixes the right-hand side of the desired dependency, while the second one asks for an FD with arbitrary RHS holding in the database. While their parameterized complexity will turn out to be the same, there are some differences in their discovery.

FUNCTIONAL DEPENDENCY*fixed RHS*

Instance: A relational database $\mathfrak{r}$ over schema R, an attribute $a \in R$,
and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R\backslash\{a\}$ with $|X| = k$ such that
the functional dependency $X \to a$ holds in $\mathfrak{r}$?

FUNCTIONAL DEPENDENCY

Instance: A relational database $\mathfrak{r}$ over schema R
and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R$ with $|X| = k$ and an attribute $a \in R\backslash X$
such that the functional dependency $X \to a$ holds in $\mathfrak{r}$?

The unparameterized variant of FUNCTIONAL DEPENDENCY*fixed RHS* is NP-

14

complete even if the number of admissible values for each attribute is at most 2 [20]. It is the same to ask for a functional dependency whose left-hand side is of size at most k; unless of course $k \geq |R|$ since then no non-trivial FD adheres to the (exact) size constraint. Again, we ignore this issue.

Recall that we say that a functional dependency $X \to a$ is minimal if its LHS X is inclusion-wise minimal among all $X' \subseteq R$ such that $X' \to a$ is valid.

ENUMERATE MINIMAL FDS*fixed RHS*

> *Instance:* A relational database $\mathfrak{r}$ over schema R and an attribute $a \in R$.

Enumeration: List all minimal, valid, non-trivial functional dependencies of $\mathfrak{r}$ with right-hand side a.

ENUMERATE MINIMAL FDS

> *Instance:* A relational database $\mathfrak{r}$.

Enumeration: List all minimal, valid, non-trivial functional dependencies of $\mathfrak{r}$.

ENUMERATE MINIMAL UCCS and ENUMERATE MINIMAL FDS can be solved in output-polynomial time (in incremental polynomial time or with polynomial delay, respectively) if and only if this is possible for the TRANSVERSAL HYPERGRAPH problem [26].

3.2. Detection

Next, we prove the parameterized complexity of the detection problems for unique column combinations and functional dependencies.

We remark that since the first announcement of our results, we learned that the UNIQUE COLUMN COMBINATION and the FUNCTIONAL DEPENDENCY*fixed RHS* problem have been studied before from a parameterized perspective, under different names and in different contexts. Therefore, some of the following results already appeared in the literature. Namely, Lemma 6 was obtained independently by Froese et al. [31], and Cotta and Moscato [16] showed the W[2]-completeness of FUNCTIONAL DEPENDENCY*fixed RHS* using different techniques. Our work can be seen as a unifying framework that shows that UNIQUE COLUMN COMBINATION is also contained in W[2] and settles the complexity of the more general FUNCTIONAL DEPENDENCY problem (with an arbitrary right-hand side) in a way that additionally sheds some light on the enumeration complexity of the related discovery problems.

$V = \{a, b, c, d, e\}$

	a	b	c	d	e
	0	0	0	0	0
$E_1 = \{a, b, d\}$	1	1	0	1	0
$E_2 = \{a, d, e\}$	2	0	0	2	2
$E_3 = \{b, d, e\}$	0	3	0	3	3
$E_4 = \{b, c\}$	0	4	4	0	0

	a	b	c	d
r^*	0	2	1	1
	1	1	1	1
	2	0	0	1
	1	2	2	0
	0	1	2	0

$\Big\}\,\mathfrak{r}$

	a	b	c	d
r^*	0	2	1	1
	1	1	1	1
	2	0	0	1
	1	2	2	0
	0	1	2	0
r_b	0	$\times$	1	1
r_c	0	2	$\times$	1
r_d	0	2	1	$\times$

$\Big\}\,\mathfrak{r}$; $\Big\}\,\mathfrak{r}'\backslash\mathfrak{r}$

(a)   (b)

Figure 3: Illustration of the reductions in Lemmas 6 and 7. (a) An instance of HITTING SET and the equivalent instance of UNIQUE COLUMN COMBINATION. (b) An instance $\mathfrak{r}$ of FUNCTIONAL DEPENDENCY$_{\text{fixed RHS}}$ with fixed right-hand side a and the equivalent instance $\mathfrak{r}'$ of FUNCTIONAL DEPENDENCY. The functional dependencies $ab \to d$ holds in $\mathfrak{r}$, but not in $\mathfrak{r}'$.

We start by proving that UNIQUE COLUMN COMBINATION is hard for $\mathsf{W}[2]$ under parameterized reductions.

Lemma 6. *There is a parameterized reduction from* HITTING SET *to the* UNIQUE COLUMN COMBINATION *problem.*

Proof. Let $(V, \mathcal{H})$ be the hypergraph given in the input to the HITTING SET problem. Without loss of generality, we can assume it to be Sperner; otherwise, we replace it by its minimization $\min(\mathcal{H})$ (in quadratic time). Observe that $\min(\mathcal{H})$ has a hitting set of size at most k iff $\mathcal{H}$ has one. We construct from $\mathcal{H}$ in polynomial time a database $\mathfrak{r}$ over schema V such that the minimal difference sets of $\mathfrak{r}$ are the edges of $\mathcal{H}$. The lemma then follows immediately from Observation 5. In particular, since Observation 5 transfers solutions, the parameter k is preserved by the reduction.

Let $E_1, E_2, \ldots, E_m$ be the edges of $\mathcal{H}$. We take the integers $0, 1, \ldots, m$ as the admissible values for all attributes in V. As rows, we first add the all-zero tuple $r_0 = (0)_{a \in V}$ to $\mathfrak{r}$. For each $i \in [m]$, we also add the record r_i defined as

$$r_i[a] = \begin{cases} i, & \text{if } a \in E_i; \\ 0, & \text{otherwise.} \end{cases}$$

See Figure 3a for an illustration. Clearly, $\mathfrak{r}$ can be computed in linear time.

Any edge E_i is a difference set in $\mathfrak{r}$, namely, that of the pair (r_0, r_i). Any other difference set must come from a pair (r_i, r_j) with $1 \leq i < j \leq m$. It is easy to see that those rows disagree in $E_i \cup E_j$, which is not minimal. Since $\mathcal{H}$ is Sperner, it contains exactly the minimal difference sets of $\mathfrak{r}$. $\qquad\square$

The next two reductions are straightforward due to the similar structures of uniques and functional dependencies. While a UCC separates any pair of rows, an FD $X \to a$ needs to distinguish only those with $r[a] \neq s[a]$.

Lemma 7. *There are parameterized reductions*

  (i) from UNIQUE COLUMN COMBINATION
      to FUNCTIONAL DEPENDENCY$_{\textit{fixed RHS}}$;

  (ii) from FUNCTIONAL DEPENDENCY$_{\textit{fixed RHS}}$
      to FUNCTIONAL DEPENDENCY.

Proof. To prove Statement (i), we add a single unique column to the database and fix it as the right-hand side of the sought functional dependency. Let $\mathfrak{r} = \{r_1, r_2, \ldots, r_{|\mathfrak{r}|}\}$ be a database over schema R, and a an attribute not previously in R. We construct $\mathfrak{r}'$ over $R \cup \{a\}$ by adding, for each r_i, the row r_i' defined by $r_i'[R] = r_i[R]$ and $r_i'[a] = i$. The reduction maps an instance $(\mathfrak{r}, R, k)$ of UNIQUE COLUMN COMBINATION to the instance $(\mathfrak{r}', R \cup \{a\}, a, k)$ of FUNCTIONAL DEPENDENCY$_{\textit{fixed RHS}}$. Since, for any two distinct rows $r_i', r_j' \in \mathfrak{r}'$, $i \neq j$, we have $r_i'[a] \neq r_j'[a]$, the left-hand sides of the non-trivial, valid FDs $X \to a$ in $\mathfrak{r}'$ are in one-to-one correspondence to the UCCs in $\mathfrak{r}$.

To reduce FUNCTIONAL DEPENDENCY$_{\textit{fixed RHS}}$ to FUNCTIONAL DEPENDENCY, we need to mask all "unwanted" FDs with RHS different from the fixed attribute a. See Figure 3b for an example. Let again $\mathfrak{r}$ be the input database over R. To construct the resulting database $\mathfrak{r}'$ over the same schema R, we take all rows from $\mathfrak{r}$ and add $|R| - 1$ new ones. Fix an arbitrary record $r^* \in \mathfrak{r}$ and let $\times$ be a new symbol that does not previously appear as a value. For each attribute $b \in R \setminus \{a\}$, we add the row r_b satisfying $r_b[R \setminus \{b\}] = r^*[R \setminus \{b\}]$ and $r_b[b] = \times$. The rows r^* and r_b now witness that any non-trivial FD $X \to b$ fails in $\mathfrak{r}'$.

It is left to prove that $X \to a$ holds in $\mathfrak{r}$ if and only if it holds in $\mathfrak{r}'$. Evidently, any valid FD in $\mathfrak{r}'$ is also valid in the subset $\mathfrak{r}$. Suppose $X \to a$ holds in $\mathfrak{r}$ and let rows $r, s \in \mathfrak{r}'$ be such that $r[a] \neq s[a]$. The only case where the conclusion $r[X] \neq s[X]$ may possibly be in doubt is if $r \in \mathfrak{r}' \setminus \mathfrak{r}$ and $s \in \mathfrak{r}$ (all new rows in $\mathfrak{r}' \setminus \mathfrak{r}$ agree on a). Hence, $r = r_b$ for some $b \neq a$. If $b \in X$,

the new value $\times$ appears in the projection $r[X]$ but not in $s[X]$; otherwise, we have $r[X] = r^*[X]$. Since $X \to a$ holds for the pair $r^*, s \in \mathfrak{r}$, the relation $r[X] = r^*[X] \neq s[X]$ follows. $\qquad\square$

The next lemma proves that every instance of the unrestricted FUNC-TIONAL DEPENDENCY problem can be expressed as an equivalent Boolean formula in conjunctive normal form. Since CNF formulas are exactly the 2-normalized ones, we obtain a reduction to WEIGHTED 2-NORMALIZED SATISFIABILITY. This is the main result of this section.

Lemma 8. *There is a parameterized reduction from* FUNCTIONAL DEPEN-DENCY *to the* WEIGHTED 2-NORMALIZED SATISFIABILITY *problem.*

Proof. Given a database $\mathfrak{r}$ over R, we derive a CNF formula that has a satisfying assignment of Hamming weight $k+1$ if and only if there is a non-trivial FD with left-hand side of size k that holds in $\mathfrak{r}$. We use two types of variables distinguished by their semantic purpose, $\mathtt{Var}_{LHS} = \{x_a \mid a \in R\}$ and $\mathtt{Var}_{RHS} = \{y_a \mid a \in R\}$. Some variable x_a from $\mathtt{Var}_{LHS}$ being set to TRUE corresponds to the attribute a appearing on the left-hand side of the sought functional dependency; for y_a from $\mathtt{Var}_{RHS}$, this means the attribute is the right-hand side.

We start with the RHS. It might be tempting to enforce that any satisfying assignment chooses exactly one variable from $\mathtt{Var}_{RHS}$. We prove below that this is not necessary and forgoing the $O(|R|^2)$ clauses representing this constraint allows for a (slightly) leaner construction. However, we do have to ensure that at least one variable from $\mathtt{Var}_{RHS}$ is set to TRUE and the corresponding one in $\mathtt{Var}_{LHS}$ is FALSE. This is done by the clauses $C_{RHS} = \bigvee_{y_a \in \mathtt{Var}_{RHS}} y_a$ and $C_a = \neg y_a \vee \neg x_a$ for each $a \in R$. The subformula φ_{RHS} is their conjunction.

We now model the LHS. For any attribute $a \in R$ and rows $r, s \in \mathfrak{r}$, let

$$C_{a,r,s} = \neg y_a \vee \bigvee_{\substack{b \in R \setminus \{a\} \\ r[b] \neq s[b]}} x_b.$$

Intuitively, the clause $C_{a,r,s}$ represents the fact that if $X \to a$ is a valid, non-trivial FD, then X has to contain at least one attribute b, different from a, such that $r[b] \neq s[b]$. From these clauses, we assemble the subformula

$$\varphi_a = \bigwedge_{\substack{r,s \in \mathfrak{r} \\ r[a] \neq s[a]}} C_{a,r,s}.$$

18

The output of our reduction is the formula $\varphi = \varphi_{RHS} \wedge \bigwedge_{a \in R} \varphi_a$. Indeed, φ is in conjunctive normal form and has $\mathrm{O}(|R||\mathfrak{r}|^2)$ clauses with at most $|R|$ literals each. An encoding of φ is computable in time linear in its size.

Regarding the correctness of the reduction, recall that we claimed φ to have a weight $k{+}1$ satisfying assignment iff a non-trivial functional dependency $X \to a$ of size k holds in $\mathfrak{r}$. Suppose that the latter is true. We show that setting the variable y_a as well as all x_b with $b \in X$ to TRUE (and all others to FALSE) satisfies φ. Note that the assignment automatically satisfies C_{RHS} and all $C_{b,r,s}$ with $b \neq a$. We are left with the subformula φ_a containing the clauses $C_{a,r,s}$ for row pairs with $r[a] \neq s[a]$. To distinguish these pairs, the LHS X includes, for each of them, some attribute $b \in R \backslash \{a\}$ such that $r[b] \neq s[b]$. Clause $C_{a,r,s}$ is then satisfied by the corresponding literal x_b.

For the other direction, we identify assignments with the variables they set to TRUE. Let $A \subseteq \mathtt{Var}_{LHS} \cup \mathtt{Var}_{RHS}$ be an assignment of Hamming weight $|A| = k{+}1$ that satisfies φ. The assignment induces two subsets of the schema R, namely, $X = \{a \in R \mid x_a \in A \cap \mathtt{Var}_{LHS}\}$ and $Y = \{a \in R \mid y_a \in A \cap \mathtt{Var}_{RHS}\}$. Due to the clause C_{RHS}, Y is non-empty and X contains at most k elements. Moreover, X and Y are disjoint as the C_a are all satisfied. We say that the generalized functional dependency $X \to Y$ holds in a database if $X \to a$ holds for every $a \in Y$. It is clearly enough to show that $X \to Y$ indeed holds in $\mathfrak{r}$.

Assume $X \to a$ fails for some $a \in Y$. This is witnessed by a pair of rows $r, s \in \mathfrak{r}$ with $r[X] = s[X]$ but $r[a] \neq s[a]$, whence the clause $C_{a,r,s}$ is present in φ. Since $y_a \in A$ is in the assignment, the literal $\neg y_a$ evaluates to FALSE. Also, as X is disjoint from the difference set $\{b \in R \backslash \{a\} \mid r[b] \neq s[b]\}$, no other literal satisfies $C_{a,r,s}$, which is a contradiction. $\qquad\square$

We have established a chain of parameterized reductions between the dependency detection problems of unique column combinations and functional dependencies. The fact that the endpoints HITTING SET and WEIGHTED 2-NORMALIZED SATISFIABILITY are both W[2]-complete shows that all of the problems are, this completes the proof of Theorem 1.

3.3. Approximation and Discovery

Our reductions have implications beyond the scope of parameterized complexity. Observe that the transformations in Lemmas 6 and 7 are computable in polynomial time, do not change the size of the vertex set/relational schema

(by more than an additive constant), and preserve approximations with respect to the solution size k or $|X|$, respectively. Also, recall that the minimization version of Hitting Set is NP-hard to approximate within a factor of $(1 - \varepsilon) \ln |V|$ for every $\varepsilon > 0$ [21]. As a consequence, the minimization versions of Unique Column Combination, Functional Dependency$_{fixed\ RHS}$, and Functional Dependency all inherit the same hardness of approximation with respect to the size $|R|$ of the schema. Previously, an approximation-preserving reduction was known only for the Unique Column Combination problem, starting from Set Cover [2].

Rather than approximating minimum solutions, we are mainly interested in the discovery of minimal dependencies in databases. Traditionally, enumeration has been studied via embedded decision problems that are different from those defined in Section 3.1. Instead, the Transversal Hypergraph problem (enumerating minimal hitting sets) has been associated with the problem of deciding for two hypergraphs $\mathcal{H}$ and $\mathcal{G}$ whether $\mathcal{G} = \text{Tr}(\mathcal{H})$ or, equivalently, $\mathcal{H} = \text{Tr}(\mathcal{G})$, called the Dual problem. Enumerate Minimal UCCs analogously corresponds to decide for a database $\mathfrak{r}$ and hypergraph $\mathcal{H}$, whether $\mathcal{H}$ consists of all minimal uniques of $\mathfrak{r}$. Intuitively, this formalizes the decision whether an enumeration algorithm has found all solutions.

Both decision problems are in coNP and it was proven by Eiter and Gottlob [26] that they are many-one equivalent. Using a lifting result by Bioch and Ibaraki [4], this shows that minimal hitting sets can be enumerated in output-polynomial time if and only if minimal UCCs can, which is the case if and only if Dual is in P. Such an equivalence is theoretically appealing and has lead to the quasi-output-polynomial upper bound on the running time of hitting set/UCC enumeration that is currently the best known [30]. The connection to enumeration has also inspired the intriguing result that Dual is likely not coNP-hard as it can be solved in polynomial time when given access to $O\left(\log^2 N / \log \log N\right)$ suitably guessed nondeterministic bits [3, 27], where N denotes the total input size of the pair $(\mathcal{H}, \mathcal{G})$. There are classes of hypergraphs for which Dual is indeed in P, see for example [8, 6, 22, 27, 48], or at least in FPT with respect to certain structural parameters [29]. The two most well-known special cases that are polynomial-time solvable are the classes of hypergraphs with constant maximum degree [22] or edge size [27], respectively. Translated to our database setting, this means that attributes can only participate in a bounded number of (minimal) difference sets or that any two rows can differ only in a bounded number of columns. Both are severe restrictions and fairly uncommon in real-world databases. For a much

more thorough overview of decision problems associated with enumeration, see the recent work by Creignou et al. [17].

Unfortunately, the approach described above holds only limited value when it comes to designing practical algorithms. Imagine an implementation of the discovery of minimal UCCs of a database $\mathfrak{r}$ via repeated checks whether the hypergraph $\mathcal{H}$ of previously found solutions is already complete. Such an algorithm is bound to use an amount of memory that is exponential in the size of $\mathfrak{r}$. This is due to the fact that some databases have exponentially many minimal solutions and the decision subroutine at least has to read all of $\mathcal{H}$. Note that such a large memory consumption is not at all necessary as there are algorithms known for TRANSVERSAL HYPERGRAPH whose space complexity is only linear in the input size [29, 44]. In fact, enumeration algorithms are often analyzed not only with respect to their running time, but also in terms of space consumption, see [9, 15]. For data profiling problems like ENUMERATE MINIMAL UCCs on the other hand, space-efficient algorithms have only recently started to received some attention [5, 6, 40].

In the following, we simplify and at the same time extend the above equivalences making them usable in practice, namely, we prove Theorem 3. It states the existence of parsimonious reductions, in both directions, that relate the input instances directly on the level of the enumeration problems, without decision problems as intermediaries. This way, we characterize the enumeration complexity of unique column combinations as well as functional dependencies, both with fixed and arbitrary right-hand side. It is worth noting that our insights on enumeration do stem from the study of decision problems while the obtained results are entirely lifted.

The reductions between the decision problems HITTING SET, UNIQUE COLUMN COMBINATION, FUNCTIONAL DEPENDENCY$_{\mathit{fixed\ RHS}}$, and FUNCTIONAL DEPENDENCY (in that order) for *minimum* dependencies described in Lemmas 6 and 7 are all built on bijective correspondences between the solutions. The running times of the reductions are polynomial and independent of the given budget k. Finally, the mappings of the solution spaces also preserve set inclusions. This means, the same input transformations applied to the discovery of *minimal* dependencies are in fact parsimonious reductions from TRANSVERSAL HYPERGRAPH to ENUMERATE MINIMAL UCCs and onward to ENUMERATE MINIMAL FDs/FDs$_{\mathit{fixed\ RHS}}$. Regarding the inverse direction, Observation 5 shows that the enumeration of UCCs is at most as hard as that of hitting sets, that is, ENUMERATE MINIMAL UCCs and TRANSVERSAL HYPERGRAPH are equivalent. It is worth noting that the

21

parsimonious reductions increase the size of the instances at most by a poly-nomial factor (usually a constant one) *in the input size only* and therefore transfer the space complexity of any enumeration algorithm.

To complete the proof of Theorem 3, we still need the following lemma that characterizes the complexity of functional dependency discovery in terms of the TRANSVERSAL HYPERGRAPH UNION problem.

Lemma 9. *The problems* ENUMERATE MINIMAL FDS *and* TRANSVERSAL HYPERGRAPH UNION *are equivalent under parsimonious reductions. Moreover, there is a parsimonious reduction from* ENUMERATE MINIMAL FDS*$_{fixed\ RHS}$ to* TRANSVERSAL HYPERGRAPH.

Proof. This proof uses techniques that already helped to establish the previous lemmas of this section. Let $\mathfrak{r}$ be a database over schema R, $a \in R$ some attribute, $r, s \in \mathfrak{r}$ two rows with $r[a] \neq s[a]$. Recall that their difference set is $D(r, s) = \{b \in R \mid r[b] \neq s[b]\}$. We define their *punctured difference set* to be $D(r, s) \backslash \{a\}$. It is implicit in the proof of Lemma 8–and easy to verify from the definition of functional dependencies–that a set $X \subseteq R \backslash \{a\}$ is the left-hand side of a valid, minimal, non-trivial FD $X \to a$ if and only if it is a minimal hitting set for the hypergraph $\mathcal{D}_a = \{D(r, s) \backslash \{a\} \mid r, s \in \mathfrak{r}; r[a] \neq s[a]\}$ of punctured difference sets.

Transforming the input database $\mathfrak{r}$ over schema $R = \{a_1, \ldots, a_{|R|}\}$ into the $|R|$ hypergraphs $\mathcal{D}_{a_1}, \ldots, \mathcal{D}_{a_{|R|}}$ is a parsimonious reduction from the ENUMERATE MINIMAL FDS problem to TRANSVERSAL HYPERGRAPH UNION. In the same fashion, fixing the desired right-hand side in the input reduces ENUMERATE MINIMAL FDS*$_{fixed\ RHS}$ to TRANSVERSAL HYPERGRAPH.

The opposite reduction is the main part of the lemma. We are given hypergraphs $\mathcal{H}_1, \ldots, \mathcal{H}_d$, without loss of generality all on the same vertex set V, and we need to compute some database $\mathfrak{r}$ such that its valid, non-trivial functional dependencies are in one-to-one correspondence with the hitting sets of the $\mathcal{H}_i$. An example can be seen in Figure 4. As the relational schema, we take the set $R = V \cup \{x_1, \ldots, x_d\}$, where the x_i are attributes not previously appearing in V. The construction of $\mathfrak{r}$ starts similarly to Lemma 6. We first add the all-zeros row r_0 (with $r_0[a] = 0$ for every $a \in R$). Let $m = \sum_{i=1}^{d} |\mathcal{H}_i|$ be the total number of edges and $E_1, E_2, \ldots, E_m$ an arbitrary numbering of them. Note that if the same set of vertices is an edge of multiple hypergraphs, it appears in the list with that multiplicity. For every edge E_j,

$$V = \{a, b, c, d, e\}$$

	a	b	c	d	e	x_1	x_2	x_3
r_0	0	0	0	0	0	0	0	0
	1	1	0	0	0	1	0	0
r_2	0	2	2	0	0	1	0	0
r_3	0	3	3	0	0	0	1	0
	0	4	0	4	4	0	1	0
	5	0	5	0	0	0	0	1
	6	0	0	6	6	0	0	1
	0	7	0	0	0	0	0	1
r_a	×	0	0	0	0	0	0	0
	0	×	0	0	0	0	0	0
	0	0	×	0	0	0	0	0
	0	0	0	×	0	0	0	0
	0	0	0	0	×	0	0	0

$$\mathcal{H}_1 = \{\{a, b\}, \{b, c\}\}$$

$$\mathcal{H}_2 = \{\{b, c\}, \{b, d, e\}\}$$

$$\mathcal{H}_3 = \{\{a, c\}, \{a, d, e\}, \{b\}\}$$

Figure 4: Illustration of Lemma 9. The three hypergraphs $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ on the vertex set V are on the left and the equivalent database $\mathfrak{r}$ is on the right. The hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ share the edge $\{b, c\}$, but this results in two rows r_2 and r_3. The corresponding transversal hypergraphs are $\mathrm{Tr}(\mathcal{H}_1) = \{\{a, c\}, \{b\}\}$, $\mathrm{Tr}(\mathcal{H}_2) = \{\{b\}, \{c, b\}, \{c, e\}\}$, and $\mathrm{Tr}(\mathcal{H}_3) = \{\{a, b\}, \{b, c, d\}, \{b, c, e\}\}$. The functional dependencies $b \to x_1$ and $b \to x_2$ indeed hold in $\mathfrak{r}$, and adding the attribute a gives $ab \to x_3$. The rows in the last block eliminate all non-trivial functional dependencies $X \to v$ with $v \in V$.

we add the following row r_j,

$$r_j[a] = \begin{cases} j, & \text{if } a \in E_j; \\ 0, & \text{if } a \in V \backslash E_j; \\ 1, & \text{if } a = x_i \text{ such that } E_j \in \mathcal{H}_i; \\ 0, & \text{otherwise.} \end{cases}$$

In other words, the subtuple $r_j[V]$ is the characteristic vector of E_j only that its non-zero entries are j instead of 1; the subtuple $r_j[\{x_1, \ldots, x_d\}]$ has exactly one 1 at the position corresponding to the hypergraph containing E_j. The remaining construction of database $\mathfrak{r}$ uses an idea of Lemma 7 (ii). Let $\times$ be a new symbol. For every vertex $v \in V$, we add the row r_v with $r_v[v] = \times$ and $r_v[a] = 0$ for all other attributes $a \in R \backslash \{v\}$. The database $\mathfrak{r}$ can be obtained in time $\mathsf{poly}(m, |V|)$.

We claim that the minimal, valid, non-trivial functional dependencies of $\mathfrak{r}$ are exactly those having the form $T \to x_i$ with $T \in \mathrm{Tr}(\mathcal{H}_i)$. The existence of a parsimonious reduction from TRANSVERSAL HYPERGRAPH UNION to ENUMERATE MINIMAL FDS easily follows from that. Let $X \subseteq R$ and $a \in R \backslash X$

be such that the FD $X \rightarrow a$ holds in $\mathfrak{r}$ and is minimal. For any $v \in V$, the rows r_0 and r_v differ only in attribute v, therefore v is not the right-hand side of any non-trivial FD, whence $a = x_i$ for some $1 \leq i \leq d$. As seen above, the set X must be a minimal transversal of the hypergraph $\mathcal{D}_{x_i} = \{D(r,s) \backslash \{x_i\} \mid r,s \in \mathfrak{r}; r[x_i] \neq s[x_i]\}$. We are left to prove that $\mathcal{D}_{x_i}$ has the same minimal transversals as $\mathcal{H}_i$. Let $r,s \in \mathfrak{r}$ be rows that differ in attribute x_i, say, $r[x_i] = 1$ and $s[x_i] = 0$. We thus have $r = r_j$ for some $1 \leq j \leq m$. The rows r and s share only the value 0, if any. Therefore,

$$D(r,s) = \begin{cases} E_j \cup \{x_i\}, & \text{if } s = r_0; \\ E_j \cup E_k \cup \{x_i, x_\ell\}, & \text{if } s = r_k \text{ for } 1 \leq k \leq m \text{ such that } E_k \in \mathcal{H}_\ell; \\ E_j \cup \{x_i, v\}, & \text{if } s = r_v \text{ for } v \in V. \end{cases}$$

In the second case, note that $\ell \neq i$ since $s[x_i] = 0$. The above implies that $\mathcal{H}_i \subseteq \mathcal{D}_{x_i}$. Moreover, all edges in $\mathcal{D}_{x_i} \backslash \mathcal{H}_i$ are supersets of ones in $\mathcal{H}_i$. The respective minimizations $\min(\mathcal{D}_{x_i}) = \min(\mathcal{H}_i)$ are thus equal and, by duality, also their transversal hypergraphs $\mathrm{Tr}(\mathcal{D}_{x_i}) = \mathrm{Tr}(\mathcal{H}_i)$ are the same. $\square$

It is known that ENUMERATE MINIMAL FDS can be solved in output-polynomial time if and only if TRANSVERSAL HYPERGRAPH can [26], this has been established along the same lines as discussed in the remarks preceding Lemma 9. Notably, Eiter and Gottlob additionally presented an alternative construction in the extended version of [26] that is *almost* parsimonious. The only condition they needed to relax is the bijection between the solution spaces. They transform a database over schema R into some hypergraph $(R^2, \mathcal{F})$ such that the majority of its minimal hitting sets indeed correspond to the functional dependencies with arbitrary right-hand side. However, $\mathcal{F}$ has some $\mathrm{O}(|R|^4)$ excess solutions (that is, polynomially many in the input size only), which do not have an FD counterpart, but are easily recognizable. We leave it as an open problem to give a fully parsimonious reduction between the problems. We have shown above that this is equivalent to encoding the hitting set information of $|R|$ different hypergraphs into a single one.

4. Inclusion Dependencies

We now discuss inclusion dependencies in relational databases. We show that their detection problem, when parameterized by the solution size, is one of the first natural problems to be complete for the class $\mathsf{W}[3]$. We do so by

proving its FPT-equivalence with the weighted satisfiability problem for a certain fragment of propositional logic. Later in Section 4.4, we transfer our results to the discovery of maximal inclusion dependencies.

4.1. Problem Definitions

A Boolean formula is *antimonotone* if it only contains negative literals. We identify a variable assignment with those variables that are set to TRUE. In the case of antimonotone formulas, this means that the satisfying assignments are closed under arbitrarily turning variables to FALSE, that is, taking subsets. The empty assignment that assigns FALSE to all variables is always satisfying. Recall that a formula is 3-normalized if it is a conjunction of disjunctions of conjunctions of literals or, equivalently, if it is a conjunctions of subformulas in disjunctive normal form (DNF). An example of an antimonotone, 3-normalized formula is

$$((\neg x_1 \wedge \neg x_2 \wedge \neg x_4) \vee (\neg x_3 \wedge \neg x_4)) \wedge ((\neg x_1 \wedge \neg x_3) \vee (\neg x_2 \wedge \neg x_5) \vee (\neg x_1 \wedge \neg x_4 \wedge \neg x_5)).$$

This formula admits satisfying assignments of Hamming weight 0, 1, and 2, but none of larger weight.

The WEIGHTED ANTIMONOTONE 3-NORMALIZED SATISFIABILITY problem is the special case of WEIGHTED 3-NORMALIZED SATISFIABILITY restricted to antimonotone formulas.

WEIGHTED ANTIMONOTONE 3-NORMALIZED SATISFIABILITY (WA3NS)

Instance: An antimonotone, 3-normalized Boolean formula φ and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Does φ admit a satisfying assignment of Hamming weight k?

By the above remark, this is the same as asking for an assignment of weight at least k. The Antimonotone Collapse Theorem of Downey and Fellows [23, 24] implies that the WA3NS special case is W[3]-complete on its own.

The inclusion-wise maximal satisfying assignments carry the full information about the collection of all satisfying assignments. It is therefore natural to define the corresponding enumeration problem as follows.

ENUMERATE MAXIMAL SATISFYING WA3NS ASSIGNMENTS

Instance: An antimonotone, 3-normalized Boolean formula φ.

Enumeration: List all maximal satisfying assignments of φ.

For inclusion dependencies, the situation is similar. Every subset of a valid IND is also valid. Asking for a dependency of size exactly k is thus the same as asking for one of size at least k. We define two variants of the decision problem, similar as we did with functional dependencies. The more restricted variant requires the two databases to have the same schema with the identity mapping between columns.

INCLUSION DEPENDENCY*Identity*

Instance: Two relational databases $\mathfrak{r}$, $\mathfrak{s}$ over schema R and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R$ with $|X| = k$ such that $\mathfrak{r}[X] \subseteq \mathfrak{s}[X]$ is an inclusion dependency?

INCLUSION DEPENDENCY

Instance: Two relational databases, $\mathfrak{r}$ over schema R and $\mathfrak{s}$ over S, and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R$ with $|X| = k$ and an injective mapping $\sigma\colon X \to S$ such that $\mathfrak{r}[X] \subseteq \mathfrak{s}[\sigma(X)]$ is an inclusion dependency?

The unparameterized variant of the general INCLUSION DEPENDENCY problem is NP-complete already for pairs of binary databases [38].

The solutions of INCLUSION DEPENDENCY*Identity* are mere subsets of the underlying schema, therefore it is clear what we mean by a maximal solution. The case of the general INCLUSION DEPENDENCY problem is slightly more intricate. Recall from Section 2.2 that we say a general inclusion dependency (X, σ) is maximal if there is no other IND (X', σ') such that $X \subsetneq X'$ is a proper subset and σ is the restriction of σ' to X. Note that the pair (X', τ) with an alternative mapping τ might still be a valid inclusion dependency.

This leads to the following enumeration problems.

ENUMERATE MAXIMAL INDS*Identity*

> *Instance:* Two relational databases $\mathfrak{r}$, $\mathfrak{s}$ over the same schema.

> *Enumeration:* List all maximal valid inclusion dependencies between $\mathfrak{r}$ and $\mathfrak{s}$ with the identity mapping between the columns.

ENUMERATE MAXIMAL INDS

> *Instance:* Two relational databases $\mathfrak{r}$ and $\mathfrak{s}$.

> *Enumeration:* List all maximal valid inclusion dependencies between $\mathfrak{r}$ and $\mathfrak{s}$.

4.2. Membership in $W[3]$

We show that both variants of the INCLUSION DEPENDENCY decision problem are contained in the class $\mathsf{W}[3]$. Recall that the INCLUSION DEPENDENCY*Identity* problem restricts the input to pairs $(\mathfrak{r}, \mathfrak{s})$ of databases over the same schema and forbids solutions in which the set of values $\mathfrak{r}[a]$ of one column are contained in $\mathfrak{s}[b]$ for some other column $b \neq a$. As a first step, we show (not entirely surprisingly) that this variant is at most as hard as the general problem.

Lemma 10. *There is a parameterized reduction from* INCLUSION DEPENDENCY*Identity* *to* INCLUSION DEPENDENCY.

Proof. Let $\mathfrak{r}$ and $\mathfrak{s}$ be two databases over the schema R and let $t^\times = (\times_a)_{a \in R}$ be a new row, where the $\times_a$ are $|R|$ different symbols none of which are previously used anywhere in $\mathfrak{r}$ or $\mathfrak{s}$. In the restricted setting, an inclusion dependency is a set $X \subseteq R$ of columns such that $\mathfrak{r}[X] \subseteq \mathfrak{s}[X]$. It is easy to see that $(\mathfrak{r}, \mathfrak{s})$ has such an inclusion dependency of size k, for any k, if and only if $(\mathfrak{r} \cup \{t^\times\}, \mathfrak{s} \cup \{t^\times\})$ has an inclusion dependency of the same size with an arbitrary mapping between the columns since $\times_a \in \mathfrak{s}[b]$ holds iff $a = b$. The lemma follows from here. $\qquad\square$

To demonstrate the membership of the general problem in $\mathsf{W}[3]$, we reduce is to WA3NS. Namely, we compute from the two databases an antimonotone, 3-normalized formula which has a weight k satisfying assignment if and only if the databases admit an inclusion dependency of that cardinality. For this, we use a correspondence between *pairs* of attributes and Boolean variables.

Lemma 11. *There is a parameterized reduction from* INCLUSION DEPEN-DENCY *to* WEIGHTED ANTIMONOTONE 3-NORMALIZED SATISFIABILITY.

Proof. Let $R = \{a_1, \ldots, a_{|R|}\}$ and $S = \{b_1, \ldots, b_{|S|}\}$ be two schemas. We introduce a Boolean variable $x_{i,j}$ for each pair of attributes $a_i \in R$ and $b_j \in S$. We let Var_P denote the set of variables corresponding to a collection $P \subseteq R \times S$ of such pairs. Consider a subset $X \subseteq R$ together with an injection $\sigma \colon X \to S$. From this, we construct a truth assignment including the variable $x_{i,j}$ (setting it to TRUE) iff $a_i \in X$ and $\sigma(a_i) = b_j$. 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 antimonotone Boolean formula φ_{map}.

$$\varphi_{map} = \left(\bigwedge_{i=1}^{|R|} \bigwedge_{j=1}^{|S|-1} \bigwedge_{j'=j+1}^{|S|} (\neg x_{i,j} \vee \neg x_{i,j'}) \right) \wedge \left(\bigwedge_{j=1}^{|S|} \bigwedge_{i=1}^{|R|-1} \bigwedge_{i'=i+1}^{|R|} (\neg x_{i,j} \vee \neg x_{i',j}) \right).$$

The first half of φ_{map} expresses that, for every pair of variables $x_{i,j}$ and $x_{i,j'}$ with $j \neq j'$, at most one of them shall be TRUE; the second half is satisfied if the same holds for all pairs $x_{i,j}$ and $x_{i',j}$ with $i \neq i'$. Conversely, a satisfying assignment A (a subset of $\text{Var}_{R \times S}$) for φ_{map} defines a relation $\sigma \subseteq R \times S$ and a set $X \subseteq R$ by setting $\sigma = \{(a_i, b_j) \mid x_{i,j} \in A\}$ and $X = \{a_i \in R \mid \exists 1 \leq j \leq |S| \colon x_{i,j} \in A\}$. By construction, the relation σ is not only a function $\sigma \colon X \to S$, but an injection. In summary, φ_{map} is fulfilled exactly by the assignments described above. Observe that φ_{map} is in conjunctive normal form and therefore also 3-normalized as each literal is a conjunctive clause of length 1.

We now formalize the requirement that a configuration (X, σ) is an inclusion dependency in a given pair of databases $\mathfrak{r}$ and $\mathfrak{s}$ over the respective schemas R and S, that is, that $\mathfrak{r}[X] \subseteq \mathfrak{s}[\sigma(X)]$ holds. First, assume that each database consists only of a single row r_ℓ and s_m, respectively. We say a pair of attributes $(a_i, b_j) \in R \times S$ is *forbidden* for r_ℓ and s_m if $r_\ell[a_i] \neq s_m[b_j]$. Let $F_{\ell,m}$ be the set of all forbidden pairs. For a configuration (X, σ) to be an IND, the variables $x_{i,j}$ need to be set to FALSE for all $(a_i, b_j) \in F_{\ell,m}$. In terms of Boolean formulas, this is represented by the conjunctive clause $M_{\ell,m} = \bigwedge_{x \in \text{Var}_{F_{\ell,m}}} \neg x$. It follows that (X, σ) is an inclusion dependency if and only if the corresponding variable assignment satisfies both φ_{map} and $M_{\ell,m}$.

Now suppose $\mathfrak{s}$ has multiple rows, while $\mathfrak{r}$ is still considered to have only one. The configuration (X, σ) is an IND for $(\mathfrak{r}, \mathfrak{s})$ iff it is one for at least one

instance $(\mathfrak{r}, \{s_m\})$ with $s_m \in \mathfrak{s}$. If also $\mathfrak{r}$ has more records, then (X, σ) is an IND for $(\mathfrak{r}, \mathfrak{s})$ iff it is one in each instance $(\{r_\ell\}, \mathfrak{s})$ with $r_\ell \in \mathfrak{r}$. Therefore, we obtain an inclusion dependency if and only if φ_{map} and the formula

$$\varphi = \bigwedge_{r_\ell \in \mathfrak{r}} \bigvee_{s_m \in \mathfrak{s}} M_{\ell,m}$$

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

The formula $\varphi \wedge \varphi_{map}$ is antimonotone and 3-normalized. The (disjunctive) clauses of φ_{map} can be constructed in total time $O(|R|^2|S| + |R||S|^2)$ and all sets $F_{\ell,m}$ together are computable in time $O(|\mathfrak{r}||\mathfrak{s}||R||S|)$. An encoding of $\varphi \wedge \varphi_{map}$ can thus be obtained from the input databases $\mathfrak{r}$ and $\mathfrak{s}$ in polynomial time. Finally, by the above observation that any solution for the sub-formula φ_{map} that corresponds to (X, σ) has weight $|X|$, the reduction preserves the parameter. $\qquad\square$

4.3. Hardness for $\mathsf{W}[3]$

We now show that detecting inclusion dependencies is also hard for $\mathsf{W}[3]$. We argue that the existence of weighted satisfying assignments for 3-normalized, antimonotone formulas can be decided by solving instances of the more restricted INCLUSION DEPENDENCY$_{Identity}$ variant. For the reduction, we make use of indicator functions. On the one hand, we interpret propositional formulas φ over n variables as Boolean functions $f_\varphi \colon \{0,1\}^n \to \{0,1\}$ in the obvious way. On the other hand, for a pair of databases $\mathfrak{r}$ and $\mathfrak{s}$ over the the same schema R, we represent any subset $X \subseteq R$ by its characteristic vector of length $|R|$. We then define the *indicator function* $f_{(\mathfrak{r},\mathfrak{s})} \colon \{0,1\}^{|R|} \to \{0,1\}$ by requiring that $f_{(\mathfrak{r},\mathfrak{s})}(X) = 1$ holds if and only if X is an inclusion dependency (with the identity mapping between the columns).

We claim that for any formula φ that is antimonotone and 3-normalized, there is a pair $(\mathfrak{r}, \mathfrak{s})$ of databases computable in polynomial time such that $f_\varphi = f_{(\mathfrak{r},\mathfrak{s})}$. Clearly, this gives a parameterized reduction from WA3NS to the INCLUSION DEPENDENCY$_{Identity}$ problem. The remainder of this section is dedicated to proving this claim. Recall that the top level connective of a 3-normalized formula is a conjunction. We start by demonstrating how to model this using databases.

Lemma 12. *Let $(\mathfrak{r}^{(1)}, \mathfrak{s}^{(1)})$ and $(\mathfrak{r}^{(2)}, \mathfrak{s}^{(2)})$ be two pairs of databases, all over the same schema R, with indicator functions $f^{(1)}$ and $f^{(2)}$, respectively. There exists a polynomial time computable pair $(\mathfrak{r}, \mathfrak{s})$ over R of size $|\mathfrak{r}| = |\mathfrak{r}^{(1)}| + |\mathfrak{r}^{(2)}|$ and $|\mathfrak{s}| = |\mathfrak{s}^{(1)}| + |\mathfrak{s}^{(2)}|$, having indicator function $f_{(\mathfrak{r},\mathfrak{s})} = f^{(1)} \wedge f^{(2)}$.*

Proof. Without loosing generality, the values appearing in $\mathfrak{r}^{(1)}$ and $\mathfrak{s}^{(1)}$ are disjoint from those in $\mathfrak{r}^{(2)}$ and $\mathfrak{s}^{(2)}$. We straightforwardly construct $(\mathfrak{r}, \mathfrak{s})$ as $\mathfrak{r} = \mathfrak{r}^{(1)} \cup \mathfrak{r}^{(2)}$ and $\mathfrak{s} = \mathfrak{s}^{(1)} \cup \mathfrak{s}^{(2)}$, which matches the requirements on both the computability and size. We still need to show $f_{(\mathfrak{r},\mathfrak{s})} = f^{(1)} \wedge f^{(2)}$.

Equivalently, we prove that a set $X \subseteq R$ is an inclusion dependency in $(\mathfrak{r}, \mathfrak{s})$ if and only if it is one in both pairs $(\mathfrak{r}^{(1)}, \mathfrak{s}^{(1)})$ and $(\mathfrak{r}^{(2)}, \mathfrak{s}^{(2)})$. Let X be an IND in $(\mathfrak{r}^{(1)}, \mathfrak{s}^{(1)})$ as well as $(\mathfrak{r}^{(2)}, \mathfrak{s}^{(2)})$. That means, for every row $r \in \mathfrak{r}^{(1)}$, there exists some $s \in \mathfrak{s}^{(1)}$ with $r[X] = s[X]$; same for $\mathfrak{r}^{(2)}$ and $\mathfrak{s}^{(2)}$. As all those rows are also present in $(\mathfrak{r}, \mathfrak{s})$, X is an IND there as well. Conversely, suppose X is not an inclusion dependency in, say, $(\mathfrak{r}^{(1)}, \mathfrak{s}^{(1)})$. Then, $\mathfrak{r}^{(1)}$ has a row r that disagrees with every $s \in \mathfrak{s}^{(1)}$ on some attribute in X. The record r is also in present in $\mathfrak{r}$ and all rows in $\mathfrak{s}$ belong either to $\mathfrak{s}^{(1)}$ or have completely disjoint values. This results in $r[X] \neq s[X]$ for *every* record $s \in \mathfrak{s}$. $\qquad \square$

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 representing conjunctions (for example, the one above). The reason is as follows. Negative literals are easily expressible by pairs of single-row databases. Together with FPT-time procedures of constructing conjunctions as well as disjunctions, one could encode antimonotone Boolean formulas of arbitrary logical depth. The Antimonotone Collaps Theorem [25], states that the WEIGHTED ANTIMONOTONE t-NORMALIZED SATISFIABILITY problem is $\mathsf{W}[t]$-complete for every odd $t \geq 3$. This would then render INCLUSION DEPENDENCY$_{Identity}$ to be hard for all classes $\mathsf{W}[t]$ and, as a consequence of Lemmas 10 and 11, the W-hierarchy would collapse to its third level. That being said, there is a method specifically tailored to antimonotone DNF formulas.

Lemma 13. *Let φ be an antimonotone formula in disjunctive normal form. There are relational databases $\mathfrak{r}$ and $\mathfrak{s}$ over the same schema computable in time polynomial in the size of φ such that $f_\varphi = f_{(\mathfrak{r},\mathfrak{s})}$.*

Proof. Let $x_1, \ldots, x_n$ be the variables of φ. Define schema $R = \{a_1, \ldots, a_n\}$ by identifying variable x_i with attribute a_i. We first describe how to obtain $\mathfrak{r}$ and subsequently construct a matching database $\mathfrak{s}$. Let $M_1, \ldots, M_m$ denote the m constituting conjunctive clauses of the DNF formula φ. For each M_j, we define the row r_j, similarly as in the proof of Lemma 6, as

$$r_j[a_i] = \begin{cases} j, & \text{if variable } x_i \text{ occurs in } M_j; \\ 0, & \text{otherwise.} \end{cases}$$

$\varphi = M_1 \vee M_2 \vee M_3$

$M_1 = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$
$M_2 = (\neg x_2 \wedge \neg x_4 \wedge \neg x_5)$
$M_3 = (\neg x_1 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_6)$

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 5: Illustration of Lemma 13. The antimonotone DNF formula φ on the left has the three conjunctive clauses M_1, M_2, M_3. The equivalent instance of INCLUSION DEPENDENCY$_{Identity}$ consists of database $\mathfrak{r}$ in the center and $\mathfrak{s}$ on the right. There are three maximal inclusion dependencies $\{a_4, a_5, a_6\}$, $\{a_1, a_3, a_6\}$, and $\{a_2, a_5\}$. Adding any more attributes to either of them would create a hitting set for the conjunctive clauses, corresponding to an unsatisfying assignment.

See Figure 5 for an example.

The second database $\mathfrak{s}$ is constructed by first creating m copies of $\mathfrak{r}$. Let $\times$ be a new symbol not appearing anywhere in $\mathfrak{r}$. In the j-th copy of $\mathfrak{r}$, we set the value for attribute a_i to $\times$ whenever x_i occurs in M_j. (See Figure 5 again.) Note that $|R| = n$ equals the number of variables of φ and $|\mathfrak{r}|$ is linear in the number m of conjunctive clauses, while $|\mathfrak{s}|$ is quadratic. The time to compute the pair $(\mathfrak{r}, \mathfrak{s})$ is linear in their combined size and polynomial in the size of φ. It is left to show that the indicator function satisfies $f_\varphi = f_{(\mathfrak{r}, \mathfrak{s})}$.

First, suppose $f_\varphi(X) = 1$ for some length-n binary vector X or, equivalently, for some subset $X \subseteq R$. We show that $f_{(\mathfrak{r}, \mathfrak{s})}(X) = 1$, meaning that X is an inclusion dependency in $(\mathfrak{r}, \mathfrak{s})$. Necessarily, we have $f_{M_j}(X) = 1$ for at least one conjunctive clause M_j. Since M_j contains only negative literals, all of its variables evaluate to FALSE. This is equivalent to X not containing any attribute that corresponds to a variable in M_j. In the j-th copy of $\mathfrak{r}$ in the database $\mathfrak{s}$, the values were changed to $\times$ for exactly those attributes. Thus, the projection $\mathfrak{s}[X]$ contains an exact copy of $\mathfrak{r}[X]$ and X is indeed an IND, resulting in $f_{(\mathfrak{r}, \mathfrak{s})}(X) = 1$.

For the opposite direction, suppose $f_\varphi(X) = 0$. Each conjunctive clause thus contains a variable corresponding to some attribute in X. Consequently, in each row of $\mathfrak{s}$, there is an attribute in X whose value was replaced by $\times$. As $\mathfrak{r}$ does not contain the symbol $\times$ at all, X is not an IND and $f_{(\mathfrak{r}, \mathfrak{s})}(X) = 0$. $\quad\square$

Lemmas 12 and 13 imply that, given an antimonotone, 3-normalized formula φ, we can build an instance $(\mathfrak{r}, \mathfrak{s})$ of INCLUSION DEPENDENCY$_{Identity}$ in FPT-time (even polynomial) such that $f_\varphi = f_{(\mathfrak{r}, \mathfrak{s})}$. Together with the findings of Section 4.2, this finishes the proof of Theorem 2.

4.4. Discovery

As we did with minimal unique column combinations and functional dependencies, we can lift our results from detecting a single inclusion dependency to discovering all of them. It turns out that there is a parsimonious equivalence with the enumeration of assignments to antimonotone, 3-normalized formulas, as detailed in Theorem 4. The key observations to prove this are once again that the reductions above are polynomial time computable, independently of the parameter, and that they preserve inclusions.

Lemmas 12 and 13 describe how to turn the formula φ in polynomial time into a pair of databases over the common schema R, which is effectively the same as $\mathtt{Var}_\varphi$, such that the inclusion dependencies $X \subseteq R$ are in canonical correspondence with the satisfying assignments $A \subseteq \mathtt{Var}_\varphi$. Moreover, the parameterized reduction preserves inclusion relations between the solutions such that the maximal dependencies also correspond to the maximal assignments. In other words, Lemmas 12 and 13 induce a parsimonious reduction from ENUMERATE MAXIMAL SATISFYING WA3NS ASSIGNMENTS to ENUMERATE MAXIMAL INDS$_{Identity}$. It is also easy to see that Lemma 10 implies a parsimonious reduction from the enumeration of such restricted inclusion dependencies to the general ENUMERATE MAXIMAL INDS problem. The lemma does nothing else but invalidating all non-identity mappings between the columns. Finally, Lemma 11 shows how to translate general inclusion dependencies back to an antimonotone, 3-normalized formula φ. Observe that the (inclusion-wise) maximal satisfying assignments of the resulting formula correspond exactly to the notion of maximality for general INDs (see Sections 2.2 and 4.1 for details). This shows the equivalence of all the enumeration problems involved. Again, the space complexity is preserved up to polynomial factors by the parsimonious reductions.

We complete the proof of Theorem 4 by showing that the problems are at least as hard as TRANSVERSAL HYPERGRAPH. This is an easy exercise using the structure of antimonotone CNFs.

Lemma 14. *The enumeration of maximal satisfying assignments of antimonotone Boolean formulas in conjunctive normal form is equivalent to* TRANSVERSAL HYPERGRAPH *under parsimonious reductions. In particular,* ENUMERATE MAXIMAL SATISFYING WA3NS ASSIGNMENTS *is at least as hard as the* TRANSVERSAL HYPERGRAPH *problem.*

Proof. For the reductions, in both directions, we identify the (disjunctive) clauses of an antimonotone CNF formula φ with the sets of variables they contain. To spell it out, let $\mathtt{Var}_\varphi$ be the set of all variables of φ and $C_1, \ldots, C_m \subseteq \mathtt{Var}_\varphi$ the clauses. Since the the formula is antimonotone, any C_i is satisfied iff there is a variable $x \in C_i$ that is assigned FALSE. In other words, an assignment $A \subseteq \mathtt{Var}_\varphi$ (the set of TRUE variables) is satisfying iff its complement $\overline{A} = \mathtt{Var}_\varphi \setminus A$ is a hitting set of the hypergraph $\{C_i\}_{i \in [m]}$. Assignment A is maximal in that regard iff $\overline{A} \in \mathrm{Tr}(\{C_i\}_i)$ is a minimal transversal. In the very same fashion, we can construct from any hypergraph $(\mathcal{H}, V)$ an antimonotone CNF formula on the variable set $\{x_v \mid v \in V\}$ by setting $\varphi = \bigwedge_{E \in \mathcal{H}} \bigvee_{v \in E} \neg x_v$. Complementing the maximal satisfying assignments of φ recovers the minimal hitting sets of $\mathcal{H}$.

The second part of the lemma follows from any CNF formula being also 3-normalized by viewing literals as conjunctive clauses of length 1. $\qquad\square$

5. Conclusion

We have determined the complexity of the detection problems for various types of multi-column dependencies, parameterized by the solution size. Our results imply that these problems do not admit FPT-algorithms unless the W-hierarchy at least partially collapses. In fact, the detection of inclusion dependencies turned out to be surprisingly hard in that it is W[3]-complete. Therefore, a small solution size alone is not enough to explain the good performance in practice. This is unfortunate as 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. Of course, our results do not preclude FPT-algorithms for *other* parameters. As an example, UNIQUE COLUMN COMBINATION on databases over schema R are trivially in FPT with respect to the parameter $|R|$ by checking all subsets. This is of course not very satisfying since assuming the schema to be small is much stronger than assuming the solutions to be small. Similarly, one could consider the maximum number d of attributes on which two rows in the data disagree as parameter. Using

the standard bounded search tree of height k with nodes of degree at most d gives an FPT-algorithm with respect to the parameter $d + k$. Again, the assumption that any two rows in a relation differ only on a few columns seems to be unrealistic for most practical data sets.

More structural research into relational databases is needed to bridge the gap between the worst-case hardness and empirical tractability of dependency detection problems. One needs to identify properties of realistic instances that explain the good performance of practical methods. In turn, such properties can then be exploited, for example, by designing multivariate algorithms with more than one parameter. This may even lead to further improvements in the running time.

On the other hand, our results regarding the discovery of all dependencies of a certain type in a database are indeed able to explain the good run times in practice. We proved that the profiling of relational data at its core is closely related to the transversal hypergraph problem. Although the exact complexity of the latter is still open, there are many empirically efficient algorithms known for it. Most importantly, modern algorithms for the enumeration of hitting sets have the advantage that their space complexity is only linear in the input size, this is a feature many data profiling algorithms are still lacking today [1, 46, 53]. Even more than large run times, prohibitive memory consumption still puts certain databases out of reach for data profiling today. The relations shown here may therefore be a way to improve the current state of the art algorithms in that direction.

Ackowledgements

References

[1] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. *Data Profiling*, volume 10 of *Synthesis Lectures on Data Management*. Morgan & Claypool Publishers, San Rafael, CA, USA, 2018. doi:10.2200/S00878ED1V01Y201810DTM052.

[2] Tatsuya Akutsu and Feng Bao. Approximating Minimum Keys and Optimal Substructure Screens. In *Proceedings of the 2nd International Conference on Computing and Combinatorics (COCOON)*, pages 290–299, 1996. `doi:10.1007/3-540-61332-3_163`.

[3] Richard Beigel and Bin Fu. Molecular Computing, Bounded Nondeterminism, and Efficient Recursion. *Algorithmica*, 25:222–238, 1999. `doi:10.1007/PL00008275`.

[4] Jan C. Bioch and Toshihide Ibaraki. Complexity of Identification and Dualization of Positive Boolean Functions. *Information and Computation*, 123:50–63, 1995. `doi:10.1006/inco.1995.1157`.

[5] Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, and Martin Schirneck. Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery. *Proceedings of the VLDB Endowment*, 13:2270–2283, 2020. `doi:10.14778/3407790.3407824`.

[6] Thomas Bläsius, Tobias Friedrich, Julius Lischeid, Kitty Meeks, and Martin Schirneck. Efficiently Enumerating Hitting Sets of Hypergraphs Arising in Data Profiling. In *Proceedings of the 21st Meeting on Algorithm Engineering and Experiments (ALENEX)*, pages 130–143, 2019. `doi:10.1137/1.9781611975499.11`.

[7] Thomas Bläsius, Tobias Friedrich, and Martin Schirneck. The Parameterized Complexity of Dependency Detection in Relational Databases. In *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 6:1–6:13, 2016. `doi:10.4230/LIPIcs.IPEC.2016.6`.

[8] Endre Boros, Vladimir Gurvich, and Peter L. Hammer. Dual Subimplicants of Positive Boolean Functions. *Optimization Methods and Software*, 10:147–156, 1998. `doi:10.1080/10556789808805708`.

[9] Florent Capelli and Yann Strozecki. Incremental Delay Enumeration: Space and Time. *Discrete Applied Mathematics*, 268:179–190, 2019. `doi:10.1016/j.dam.2018.06.038`.

[10] Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. Inclusion Dependencies and Their Interaction with Functional Dependencies. In *Proceedings of the 1st Symposium on Principles of Database Systems (PODS)*, pages 171–176, 1982. `doi:10.1145/588111.588141`.

[11] Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. On the Complexity of Solution Extension of Optimization Problems. *Theoretical Computer Science*, 2021. In press. `doi:10.1016/j.tcs.2021.10.017`.

[12] Surajit Chaudhuri and Kyuseok Shim. Including Group-By in Query Optimization. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 354–366, 1994. URL: `http://www.vldb.org/conf/1994/P354.PDF`.

[13] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong Computational Lower Bounds via Parameterized Complexity. *Journal of Computer and System Sciences*, 72:1346–1367, 2006. `doi:10.1016/j.jcss.2006.04.007`.

[14] 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`.

[15] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. Sublinear-Space and Bounded-Delay Algorithms for Maximal Clique Enumeration in Graphs. *Algorithmica*, 82:1547–1573, 2020. `doi:10.1007/s00453-019-00656-8`.

[16] Carlos Cotta and Pablo Moscato. The k-Feature Set Problem is $W[2]$-complete. *Journal of Computer and System Sciences*, 67:686–690, 2003. `doi:10.1016/S0022-0000(03)00081-3`.

[17] Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, and Heribert Vollmer. A Complexity Theory for Hard Enumeration Problems. *Discrete Applied Mathematics*, 268:191–209, 2019. `doi:10.1016/j.dam.2019.02.025`.

[18] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh.

Parameterized Algorithms. Springer, Cham, Switzerland, 2015. `doi: 10.1007/978-3-319-21275-3`.

[19] Christopher J. Date. *An Introduction to Database Systems.* Addison-Wesley Longman Publishing, Boston, MA, USA, 8th edition edition, 2003.

[20] Scott Davies and Stuart Russell. NP-Completeness of Searches for Smallest Possible Feature Sets. In *AAAI Symposium on Intelligent Relevance*, 1994. Technical Report. URL: `https://www.aaai.org/Papers/Symposia/Fall/1994/FS-94-02/FS94-02-011.pdf`.

[21] Irit Dinur and David Steurer. Analytical Approach to Parallel Repetition. In *Proceedings of the 46th Symposium on Theory of Computing (STOC)*, page 624–633, 2014. `doi:10.1145/2591796.2591884`.

[22] Carlos Domingo, Nina Mishra, and Leonard Pitt. Efficient Read-Restricted Monotone CNF/DNF Dualization by Learning with Membership Queries. *Machine Learning*, 37:89–110, 1999. `doi:10.1023/A:1007627028578`.

[23] Rodney G. Downey and Michael R. Fellows. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal on Computing*, 24:873–921, 1995. `doi:10.1137/S0097539792228228`.

[24] 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. `doi:10.1016/0304-3975(94)00097-3`.

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

[26] Thomas Eiter and Georg Gottlob. Identifying the Minimal Transversals of a Hypergraph and Related Problems. *SIAM Journal on Computing*, 24:1278–1304, 1995. `doi:10.1137/S0097539793250299`.

[27] Thomas Eiter, Georg Gottlob, and Kazuhisa Makino. New Results on Monotone Dualization and Generating Hypergraph Transversals. *SIAM Journal on Computing*, 32:514–537, 2003. `doi:10.1137/S009753970240639X`.

[28] Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. Computational Aspects of Monotone Dualization: A Brief Survey. *Discrete Applied Mathematics*, 156:2035–2049, 2008. `doi:10.1016/j.dam.2007.04.017`.

[29] Khaled M. Elbassioni, Matthias Hagen, and Imran Rauf. Some Fixed-Parameter Tractable Classes of Hypergraph Duality and Related Problems. In *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation, (IWPEC)*, pages 91–102, 2008. `doi:10.1007/978-3-540-79723-4_10`.

[30] Michael L. Fredman and Leonid G. Khachiyan. On the Complexity of Dualization of Monotone Disjunctive Normal Forms. *Journal of Algorithms*, 21:618–628, 1996. `doi:10.1006/jagm.1996.0062`.

[31] Vincent Froese, René van Bevern, Rolf Niedermeier, and Manuel Sorge. Exploiting Hidden Structure in Selecting Dimensions That Distinguish Vectors. *Journal of Computer and System Sciences*, 82:521–535, 2016. `doi:10.1016/j.jcss.2015.11.011`.

[32] Andrew Gainer-Dewar and Paola Vera-Licona. The Minimal Hitting Set Generation Problem: Algorithms and Computation. *SIAM Journal on Discrete Mathematics*, 31:63–100, 2017. `doi:10.1137/15M1055024`.

[33] Chris Giannella, Mehmet M. Dalkilic, Dennis P. Groth, and Edward L. Robertson. Improving Query Evaluation with Approximate Functional Dependency Based Decompositions. In *Proceedings of the 19th British National Conference on Databases, (BNCOD)*, volume 2405, pages 26–41, 2002. `doi:10.1007/3-540-45495-0_3`.

[34] Miika Hannula, Bor-Kuan Song, and Sebastian Link. An Algorithm for the Discovery of Independence from Data. *CoRR*, abs/2101.02502, 2021. ArXiv preprint. URL: `https://arxiv.org/abs/2101.02502`, `arXiv:2101.02502`.

[35] Ihab F. Ilyas, Volker Markl, Peter J. Haas, Paul Brown, and Ashraf Aboulnaga. CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In *Proceedings of the 2004 International Conference on Management of Data (SIGMOD)*, pages 647–658, 2004. `doi:10.1145/1007568.1007641`.

[36] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of *k*-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001. `doi:10.1006/jcss.2000.1727`.

[37] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Science*, 63:512–530, 2001. `doi:10.1006/jcss.2001.1774`.

[38] 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`.

[39] 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`.

[40] Henning Köhler, Uwe Leck, Sebastian Link, and Xiaofang Zhou. Possible and Certain Keys for SQL. *The VLDB Journal*, 25:571–596, 2016. `doi:10.1007/s00778-016-0430-9`.

[41] Jan Kossmann, Thorsten Papenbrock, and Felix Naumann. Data Dependencies for Query Optimization: A Survey. *The VLDB Journal*, 30, 2021. `doi:10.1007/s00778-021-00676-3`.

[42] David Maier. *The Theory of Relational Databases*. Computer Science Press, Rockville, MD, USA, 1983.

[43] Heikki Mannila and Kari-Jouko Räihä. Dependency Inference. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB)*, pages 155–158, 1987. URL: `https://www.vldb.org/conf/1987/P155.PDF`.

[44] Keisuke Murakami and Takeaki Uno. Efficient Algorithms for Dualizing Large-scale Hypergraphs. *Discrete Applied Mathematics*, 170:83–94, 2014. `doi:10.1016/j.dam.2014.01.012`.

[45] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University

Press, Oxford, UK, 2006. `doi:10.1093/acprof:oso/9780198566076.001.0001`.

[46] Thorsten Papenbrock and Felix Naumann. A Hybrid Approach to Functional Dependency Discovery. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, pages 821–833, 2016. `doi:10.1145/2882903.2915203`.

[47] Glenn N. Paulley and Per-Åke Larson. Exploiting Uniqueness in Query Optimization. In *Proceedings of the 10th International Conference on Data Engineering (ICDE)*, pages 68–79, 1994. `doi:10.1109/ICDE.1994.283003`.

[48] Uri N. Peled and Bruno Simeone. A $O(nm)$-Time Algorithm for Computing the Dual of a Regular Boolean Function. *Discrete Applied Mathematics*, 49:309–323, 1994. `doi:10.1016/0166-218X(94)90215-1`.

[49] Taneli Pusa, Mariana Galvao Ferrarini, Ricardo Andrade, Arnaud Mary, Alberto Marchetti-Spaccamela, Leen Stougie, and Marie-France Sagot. MOOMIN - mathematical exploration of 'omics data on a metabolic network. *Bioinformatics*, 36:514–523, 2020. `doi:10.1093/bioinformatics/btz584`.

[50] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–95, 1987. `doi:10.1016/0004-3702(87)90062-2`.

[51] Andrzej Skowron and Cecylia Rauszer. volume 11 of *Theory and Decision Library - Series D: System Theory, Knowledge Engineering and Problem Solving*, chapter The Discernibility Matrices and Functions in Information Systems, pages 331–362. Kluwer Academic Publishers, Dordrecht, Netherlands, 1992. `doi:10.1007/978-94-015-7975-9_21`.

[52] Aaron Swartz. MusicBrainz: A Semantic Web Service. *IEEE Intelligent Systems*, 17:76–77, 2002. See `www.musicbrainz.org`. `doi:10.1109/5254.988466`.

[53] Ziheng Wei, Uwe Leck, and Sebastian Link. Discovery and Ranking of Embedded Uniqueness Constraints. *Proceedings of the VLDB Endowment*, 12:2339–2352, 2019. `doi:10.14778/3358701.3358703`.