Computing Optimal Repairs for Functional Dependencies

ESTER LIVSHITS and BENNY KIMELFELD, Technion–Israel Institute of Technology, Israel
SUDEEPA ROY, Duke University, USA

We investigate the complexity of computing an optimal repair of an inconsistent database, in the case where integrity constraints are Functional Dependencies (FDs). We focus on two types of repairs: an optimal subset repair (optimal S-repair), which is obtained by a minimum number of tuple deletions, and an optimal update repair (optimal U-repair), which is obtained by a minimum number of value (cell) updates. For computing an optimal S-repair, we present a polynomial-time algorithm that succeeds on certain sets of FDs and fails on others. We prove the following about the algorithm. When it succeeds, it can also incorporate weighted tuples and duplicate tuples. When it fails, the problem is NP-hard and, in fact, APX-complete (hence, cannot be approximated better than some constant). Thus, we establish a dichotomy in the complexity of computing an optimal S-repair. We present general analysis techniques for the complexity of computing an optimal U-repair, some based on the dichotomy for S-repairs. We also draw a connection to a past dichotomy in the complexity of finding a "most probable database" that satisfies a set of FDs with a single attribute on the left-hand side; the case of general FDs was left open, and we show how our dichotomy provides the missing generalization and thereby settles the open problem.

CCS Concepts: • **Information systems → Inconsistent data**; **Data cleaning**; • **Theory of computation → Incomplete, inconsistent, and uncertain databases**;

Additional Key Words and Phrases: Inconsistent databases, database cleaning, optimal repairs, cardinality repairs, value repairs, functional dependencies, dichotomy, approximation

ACM Reference format:
Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2020. Computing Optimal Repairs for Functional Dependencies. *ACM Trans. Database Syst.* 45, 1, Article 4 (February 2020), 46 pages.
https://doi.org/10.1145/3360904

1 INTRODUCTION

Database inconsistency arises in a variety of scenarios and for different reasons. Data may be collected from imprecise sources (social encyclopedias/networks, sensors attached to appliances, cameras, etc.) via imprecise procedures (natural-language processing, signal processing, image analysis, etc.). Inconsistency may also arise when integrating databases of different organizations

The work of Benny Kimelfeld and Ester Livshits was supported by the Israel Science Foundation (ISF) Grant No. 1295/15. The work of Ester Livshits was also supported by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel Cyber Bureau. The work of Sudeepa Roy was supported by NSF Awards No. IIS-1552538 and No. IIS-1703431, and NIH Award No. 1R01EB025021-01.
Authors' addresses: E. Livshits and B. Kimelfeld, Computer Science Department, Technion, Haifa 3200003, Israel; emails: {esterliv, bennyk}@cs.technion.ac.il; S. Roy, 308 Research Drive, Department of Computer Science, Duke University, LSRC Building, Campus Box 90129, Durham, NC 27708, USA; email: sudeepa@cs.duke.edu.

0362-5915/2020/02-ART4 $15.00
https://doi.org/10.1145/3360904

with conflicting information, or even consistent information in conflicting formats. Arenas et al. [5] introduced a principled approach to managing inconsistency via the notions of *repairs* and *consistent query answering*. An *inconsistent database* is a database D that violates an underlying collection of integrity constraints, a *repair* is a consistent database D' that is obtained from D through a minimal sequence of editing operations, and the *consistent answers* to a query are the answers found in every repair D'.

Instantiations of the repair framework differ in their definitions of *integrity constraints*, *operations*, and *minimality* [1]. Common types of constraints are denial constraints [25] that include the classic Functional Dependencies (FDs), and inclusion dependencies [15] that include the referential (foreign-key) constraints. An operation can be a *deletion* of a tuple, an *insertion* of a tuple, and an *update* of an attribute (cell) value. Minimality can be either *local*—no strict subset of the operations achieves consistency, or *global*—no smaller (or cheaper) subset achieves consistency. For example, if only tuple deletions are allowed, then a *subset repair* [16] corresponds to a local minimum (restoring any deleted tuple causes inconsistency) and a *cardinality repair* [38] corresponds to a global minimum (consistency cannot be gained by fewer tuple deletions). The *cost* of operations may differ between tuples; this can represent different levels of trust that we have in the tuples [33, 38].

In this article, we focus on global minima under FDs via tuple deletions and value updates. Each tuple is associated with a weight that determines the cost of its deletion or a change of a single value. We study the complexity of computing a minimum repair in two settings. In the first setting, only tuple deletions are allowed, and therefore, we seek the weighted version of the cardinality repair, also referred to as *weight-maximally consistent subset* [22] and *maximum weight repairs* [14]. In the second setting, only value updates are allowed, that is, we seek what Kolahi and Lakshmanan [33] refer to as an "optimum V-repair." We refer to the two challenges as computing an optimal subset repair (optimal S-repair) and computing an optimal update repair (optimal U-repair), respectively. These problems were recently shown to be special cases of the *most likely intention* problem in the framework of probabilistic unclean databases [40] that establishes the theoretical basis for the HoloClean data cleaning system [39]. In this model, an unclean database is viewed as a result of a two-step process, similarly to the noisy-channel model: an intended clean database ("intention") is first generated, and then noise is introduced.

Indeed, the importance of computing an optimal repair arises in the challenge of *data cleaning* [24]—eliminate errors and dirt (manifested as inconsistencies) from the database. Specifically, our motivation is twofold. The obvious motivation is in fully automated cleaning, where an optimal repair is the best candidate, assuming that the system is aware of only the constraints and tuple weights. The second motivation comes from the more realistic practice of iterative, human-in-the-loop cleaning [6, 10, 19, 26]. As proposed by Bertossi [11, 12], in such systems the cost of an optimal repair can serve as a *measure of inconsistency*, i.e., an educated estimate for the extent to which the database is dirty and, consequently, the amount of effort needed for the completion of cleaning. Livshits et al. [35] showed that this measure stands out among other inconsistency measures for progress estimation in data cleaning systems, as it satisfies some very natural properties for such measures. A minimum-distance measure was also introduced by Grant et al. [27] as a measure of inconsistency for knowledge bases.

As our integrity constraints are FDs, it suffices to consider a database with a single relation, which we call here a *table*. In a general database, our results can be applied to each relation individually. A table T conforms to a relational schema $R(A_1, \ldots, A_k)$ where each A_i is an attribute. Integrity is determined by a set Δ of FDs. Our complexity analysis focuses primarily on *data complexity*, where $R(A_1, \ldots, A_k)$ and Δ are considered fixed and only T is considered input. Hence, we have infinitely many optimization problems, one for each combination of $R(A_1, \ldots, A_k)$ and Δ.

Table records have identifiers, as we wish to be able to determine easily which cells are updated in a repair. In particular, we allow duplicate tuples (with distinct identifiers).

All of the negative results in this article hold even for the more restricted case where the table T is duplicate-free and all tuples have a unit weight, and all of the positive results hold even for the more general case where we allow duplicates and distinct weights. Hence, all of our results apply to any possible combination of these two properties.

We begin with the problem of computing an optimal S-repair. The problem is known to be computationally hard for denial constraints [38]. As we discuss later, complexity results can be inferred from prior work [28] for FDs with a single attribute on the left-hand side (lhs). For general FDs, we present the algorithm OptSRepair (Algorithm 1). The algorithm seeks opportunities for simplifying the problem by eliminating attributes and FDs, until no FDs are left (and then the problem is trivial). For example, if all FDs share an attribute A on the lhs, then we can partition the table according to A and solve the problem separately on each partition; but now, we can ignore A in each partition. We refer to this simplification as "common lhs." Two additional simplifications are the "consensus" and "lhs marriage." Importantly, the algorithm terminates in polynomial time, even under *combined* complexity (where both the table and FDs are given as input).

However, OptSRepair may fail by reaching a nonempty set of FDs where no simplification can be applied. We prove two properties of the algorithm. The first is *soundness*—if the algorithm succeeds, then it returns an optimal S-repair. More interesting is the property of *completeness*—if the algorithm fails, then the problem is NP-hard. In fact, in this case the problem is APX-complete, that is, for some $\alpha > 1$ it is NP-hard to find a consistent subset with a cost lower than α times the minimum, but *some* α' is achievable in polynomial time. More so, the problem remains APX-complete if we assume that the table does not contain duplicates, and all tuples have a unit weight (in which case we say that T is *unweighted*). Consequently, we establish the following dichotomy in complexity for the space of combinations of schemas $R(A_1, \ldots, A_k)$ and FD sets Δ.

- If we can eliminate all nontrivial FDs in Δ with the three simplifications, then an optimal S-repair can be computed in polynomial time using OptSRepair.
- Otherwise, computing an optimal S-repair is APX-complete, even for unweighted tables without duplicates.

We then continue to the problem of computing an optimal U-repair. Here we do not establish a full dichotomy, but we make a substantial progress. We have found that proving hardness results for U-repairs is far more subtle than for deletions. We identify conditions where the complexity of computing an optimal U-repair and that of computing an optimal S-repair coincide. One such condition is the common lhs (i.e., all FDs share an lhs attribute). Hence, in this case, our dichotomy for S-repairs provides the precise test of tractability. We also show decomposition techniques that extend the opportunities of using the dichotomy. As an example, consider the schema Purchase(product, price, buyer, email, address) and $\Delta_0 = \{\text{product} \rightarrow \text{price}, \text{buyer} \rightarrow \text{email}\}$. We can decompose this problem into $\Delta_1 = \{\text{product} \rightarrow \text{price}\}$ and $\Delta_2 = \{\text{buyer} \rightarrow \text{email}\}$, and consider each Δ_i, for $i = 1, 2$, independently. The complexity of each Δ_i is the same in both variants of optimal repairs, and so, polynomial time. Yet, these results do not cover all sets of FDs. For example, let $\Delta_3 = \{\text{email} \rightarrow \text{buyer}, \text{buyer} \rightarrow \text{address}\}$. Kolahi and Lakshmanan [33] proved that under Δ_3, computing an optimal U-repair is NP-hard. Our dichotomy shows that it is also NP-hard (and also APX-complete) to compute an S-repair under Δ_3. Yet, this FD set does not fall in our coincidence cases.

Finally, we consider approximate repairing. For the case of an optimal S-repair, the problem easily reduces to that of *weighted vertex cover*, and hence, we get a polynomial-time 2-approximation due to Bar-Yehuda and Even [8]. To approximate optimal U-repairs, we show an efficient

reduction to S-repairs, where the loss in approximation is linear in the number of attributes. Hence, we obtain a constant-ratio approximation, where the constant has a linear dependence on the number of attributes. Kolahi and Lakshmanan [33] also gave an approximation for optimal U-repairs, but their worst-case approximation can be quadratic in the number of attributes. We show an infinite sequence of FD sets where this gap is actually realized. However, we also show an infinite sequence where our approximation is linear in the number of attributes, but theirs remains constant. Hence, in general, the two approximations are incomparable, and we can combine the two by running both approximations and taking the best.

Stepping outside the framework of repairs, a different approach to data cleaning is *probabilistic* [4, 28, 39]. The idea is to define a probability space over possible clean databases, where the probability of a database is determined by the extent to which it satisfies the integrity constraints. The goal is to find a *most probable database* that, in turn, serves as the clean outcome. As an instantiation, Gribkoff, Van den Broeck, and Suciu [28] identify probabilistic cleaning as the "Most Probable Database" problem (MPD): given a tuple-independent probabilistic database [20, 42] and a set of FDs, find the most probable database among those satisfying the FDs (or, put differently, condition the probability space on consistency). They show a dichotomy for *unary* FDs, that is, FDs with a single attribute on the left-hand side. The case of general (not necessarily unary) FDs has been left open. It turns out that there are reductions from MPD to computing an optimal S-repair and vice versa. Consequently, we are able to generalize their dichotomy to all FDs, and hence, fully settle the open problem.

This article extends a conference publication of the authors [37]. Compared to the conference version, the extension is as follows. First, we have strengthened the dichotomy in the complexity of MPD with hardness of approximation on the negative side (Theorem 3.8). Second, we have added all the proofs and intermediate results that were excluded from the conference paper. In particular, Section 4 is new and contains the full proof of our main result—the dichotomy in the complexity for S-repairs (Theorem 3.4). Moreover, we have added the proofs of Theorems 5.2, 5.5, 5.12, 5.14, and 5.15 in Section 5.

The rest of the article is organized as follows. In Section 2, we give the basic definitions and problem statements. We study the problem of computing an optimal S-repair in Section 3, where we also discuss the connection to MPD. Then, we give the full proof of our dichotomy for S-repairs in Section 4. In Section 5, we study the problem of computing an optimal U-repair. We conclude and discuss future directions in Section 6.

2 PRELIMINARIES

We first present some basic terminology and notation that we use throughout the article.

2.1 Schemas and Tables

An instance of our data model is a single table where each tuple is associated with an *identifier* and a *weight* that states how costly it is to change or delete the tuple. Such a table corresponds to a *relation schema* that we denote by $R(A_1, \ldots, A_k)$, where R is the *relation name* and $A_1, \ldots, A_k$ are distinct *attributes*. We say that $R(A_1, \ldots, A_k)$ is *k-ary*, since it has k attributes. When there is no risk of confusion, we may refer to $R(A_1, \ldots, A_k)$ by simply R.

We use capital letters from the beginning of the English alphabet (e.g., A, B, C), possibly with subscripts and/or superscripts, to denote individual attributes, and capital letters from the end of the English alphabet (e.g., X, Y, Z), possibly with subscripts and/or superscripts, to denote *sets* of attributes. We follow the convention of avoiding commas and curly braces when writing sets of attributes (e.g., ABC instead of $\{A, B, C\}$).

id	facility	room	floor	city	w
1	HQ	322	3	Paris	2
2	HQ	322	30	Madrid	1
3	HQ	122	1	Madrid	1
4	Lab1	B35	3	London	2

(a) Table T

id	facility	room	floor	city	w
2	HQ	322	30	Madrid	1
3	HQ	122	1	Madrid	1
4	Lab1	B35	3	London	2

(b) Consistent subset S_1

id	facility	room	floor	city	w
1	HQ	322	3	Paris	2
4	Lab1	B35	3	London	2

(c) Consistent subset S_2

id	facility	room	floor	city	w
3	HQ	122	1	Madrid	1
4	Lab1	B35	3	London	2

(d) Consistent subset S_3

id	facility	room	floor	city	w
1	F01	322	3	Paris	2
2	HQ	322	30	Madrid	1
3	HQ	122	1	Madrid	1
4	Lab1	B35	3	London	2

(e) Consistent update U_1

id	facility	room	floor	city	w
1	HQ	322	3	Paris	2
2	HQ	322	3	Paris	1
3	HQ	122	1	Paris	1
4	Lab1	B35	3	London	2

(f) Consistent update U_2

id	facility	room	floor	city	w
1	HQ	322	30	Madrid	2
2	HQ	322	30	Madrid	1
3	HQ	122	1	Madrid	1
4	Lab1	B35	3	London	2

(g) Consistent update U_3

Fig. 1. For Office(facility, room, floor, city) and FDs facility → city and facility room → floor, a table T, consistent subsets S_1, S_2 and S_3, and consistent updates U_1, U_2 and U_3. Changed values are marked in yellow.

We assume a countably infinite domain Val of attribute values. By a *tuple*, we mean a sequence of values in Val. A *table* T over $R(A_1, \ldots, A_k)$ has a collection $ids(T)$ of (tuple) identifiers and it maps every identifier $i \in ids(T)$ to a tuple in Val^k and a positive (non-zero) weight; we denote this tuple by $T[i]$ and this weight by $w_T(i)$. We denote by $T[*]$ the set of all tuples of T. We say that T is:

- *duplicate free* if distinct tuples disagree on at least one attribute, that is, we have $T[i] \neq T[j]$ whenever $i \neq j$;
- *unweighted* if all tuple weights are equal, that is, $w_T(i) = w_T(j)$ for all identifiers i and j.

We use $|T|$ to denote the number of tuple identifiers of T, that is, $|T| \stackrel{\text{def}}{=} |ids(T)|$. Let $\mathbf{t} = (a_1, \ldots, a_k)$ be a tuple of T. We use $\mathbf{t}.A_j$ to refer to the value a_j. If $X = A_{i_1}, \ldots, A_{i_\ell}$ is a sequence of attributes in $\{A_1, \ldots, A_k\}$, then $\mathbf{t}[X]$ denotes the tuple $(\mathbf{t}.A_{i_1}, \ldots, \mathbf{t}.A_{i_\ell})$.

Example 2.1. Our running example is based on the tables of Figure 1, over the schema Office(facility, room, floor, city), describing the location of offices in an organization. For example, the tuple $T[1]$ corresponds to an office in room 322, in the third floor of the headquarters (HQ) building, located in Paris. The meaning of the yellow background color will be clarified later. The identifier of each tuple is shown on the leftmost (gray shaded) column, and its weight on the rightmost column (also gray shaded). Note that table S_2 is duplicate free and unweighted, table S_1 is duplicate free but not unweighted, and table U_2 is neither duplicate free nor unweighted.

2.2 Functional Dependencies (FDs)

Let $R(A_1, \ldots, A_k)$ be a schema. As usual, an FD (over R) is an expression of the form $X \rightarrow Y$ where X and Y are sequences of attributes of R. We refer to X as the *left-hand side*, or *lhs*, and to Y as the *right-hand side*, or *rhs*. A table T *satisfies* $X \rightarrow Y$ if every two tuples that agree on X also agree on Y; that is, for all $\mathbf{t}, \mathbf{s} \in T[*]$, if $\mathbf{t}[X] = \mathbf{s}[X]$ then $\mathbf{t}[Y] = \mathbf{s}[Y]$. We say that T *satisfies* a set Δ of FDs if T satisfies each FD in Δ; otherwise, T *violates* Δ.

An FD $X \rightarrow Y$ is *entailed by* Δ, denoted $\Delta \models X \rightarrow Y$, if every table T that satisfies Δ also satisfies the FD $X \rightarrow Y$. The *closure* of Δ, denoted $cl(\Delta)$, is the set of all FDs that are entailed by Δ. The *closure* of an attribute set X (*w.r.t.* Δ), denoted $cl_\Delta(X)$, is the set of all attributes A such that the FD $X \rightarrow A$ is entailed by Δ. Two sets Δ_1 and Δ_2 of FDs are *equivalent* if they have the same closure (or in other words, each FD in Δ_1 is entailed by Δ_2 and vice versa, or put differently, every table that satisfies one also satisfies the other). An FD $X \rightarrow Y$ is *trivial* if $Y \subseteq X$; otherwise, it is *nontrivial*. Note that a trivial FD belongs to the closure of every set of FDs (including the empty one). We say that Δ is *trivial* if Δ does not contain any nontrivial FDs (e.g., it is empty); otherwise, Δ is *nontrivial*.

Next, we give some non-standard notation that we need for this article. A *common lhs* of an FD set Δ is an attribute A such that $A \in X$ for all FDs $X \rightarrow Y$ in Δ. An FD set Δ is a *chain* if for every two FDs $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$ it is the case that $X_1 \subseteq X_2$ or $X_2 \subseteq X_1$. Livshits and Kimelfeld [36] proved that the class of chain FD sets consists of precisely the FD sets in which the *subset repairs*, which we define in Section 2.3, can be counted in polynomial time (assuming P $\neq$ #P). The chain FD sets will arise in this work as well.

Example 2.2. In our running example (Figure 1) the set Δ consists of the following FDs:

- facility $\rightarrow$ city: a facility belongs to a single city.
- facility room $\rightarrow$ floor: a room in a facility does not extend beyond one floor.

Note that the FDs allow for the same room number to occur in different facilities (possibly on different floors, in different cities). The attribute facility is a common lhs. Moreover, Δ is a chain FD set, since $\{\text{facility}\} \subseteq \{\text{facility}, \text{room}\}$. Table T (Figure 1(a)) violates Δ, and the other tables (Figures 1(b)–1(g)) satisfy Δ.

An FD $X \rightarrow Y$ might be such that X is empty, and then we denote it by $\emptyset \rightarrow Y$ and call it a *consensus* FD. Satisfying the consensus FD $\emptyset \rightarrow Y$ means that all tuples agree on Y, or in other words, the column that corresponds to each attribute in Y consists of copies of the same value. For example, $\emptyset \rightarrow$ city means that all tuples have the same city. A *consensus attribute* (*of* Δ) is an attribute in $cl_\Delta(\emptyset)$, that is, an attribute A such that $\emptyset \rightarrow A$ is implied by Δ. We say that Δ is *consensus free* if it has no consensus attributes.

2.3 Repairs

Let $R(A_1, \ldots, A_k)$ be a schema, and let T be a table. A *subset* of T is a table S that is obtained from T by eliminating tuples. More formally, table S is a subset of T if $ids(S) \subseteq ids(T)$, and for all $i \in ids(S)$, we have $S[i] = T[i]$ and $w_S(i) = w_T(i)$. If S is a subset of T, then the *distance* from S to T, denoted $dist_{\text{sub}}(S, T)$, is the weighted sum of the tuples missing from S; that is,

$$dist_{\text{sub}}(S, T) \stackrel{\text{def}}{=} \sum_{i \in ids(T) \setminus ids(S)} w_T(i) \,.$$

A *value update* of T (or just *update* of T) is a table U that is obtained from T by changing attribute values. More formally, a table U is an update of T if $ids(U) = ids(T)$, and for all $i \in ids(U)$, we have $w_U(i) = w_T(i)$. We adopt the definition of Kolahi and Lakshmanan [33] for the distance from U to T. Specifically, if $\mathbf{u}$ and $\mathbf{t}$ are tuples of tables over R, then the *Hamming distance* $H(\mathbf{u}, \mathbf{t})$ is

the number of attributes in which $\mathbf{u}$ and $\mathbf{t}$ disagree, that is, $H(\mathbf{u}, \mathbf{t}) = |\{j \mid \mathbf{u}.A_j \neq \mathbf{t}.A_j\}|$. If U is an update of T, then the *distance* from U to T, denoted $dist_{\mathrm{upd}}(U, T)$, is the weighted Hamming distance between U and T (where every changed value counts as the weight of the tuple); that is,

$$dist_{\mathrm{upd}}(U, T) \stackrel{\mathrm{def}}{=} \sum_{i \in ids(T)} w_T(i) \cdot H(T[i], U[i]) \,.$$

Let $R(A_1, \ldots, A_k)$ be a schema, let T be table, and let Δ be a set of FDs. A *consistent subset* (*of* T *w.r.t.* Δ) is a subset S of T such that $S \models \Delta$, and a *consistent update* (*of* T *w.r.t.* Δ) is an update U of T such that $U \models \Delta$. A *subset repair*, or *S-repair*, is a consistent subset that becomes inconsistent whenever any deleted tuple is brought back. An *update repair*, or *U-repair*, is a consistent update that becomes inconsistent if any updated value is restored to its original value in T. An *optimal subset repair* of T, or just *optimal S-repair*, is a consistent subset S of T such that $dist_{\mathrm{sub}}(S, T)$ is minimal among all consistent subsets of T. Similarly, an *optimal update repair* of T, or just *optimal U-repair*, is a consistent update U of T such that $dist_{\mathrm{upd}}(U, T)$ is minimal among all consistent updates of T. When there is risk of ambiguity, we may stress that the optimal S-repair (or U-repair) is *of* T and *under* Δ or *under* R *and* Δ. Every optimal S-repair (respectively, U-repair) is an S-repair (respectively, U-repair), but not necessarily vice versa. Observe that a consistent subset (respectively, update) can be transformed into a (not necessarily optimal) S-repair (respectively, U-repair), with no increase of distance, in polynomial time.

We also define *approximations* of optimal repairs in the natural ways, as follows. For a number $\alpha \geq 1$, an α-optimal S-repair is an S-repair S of T such that $dist_{\mathrm{sub}}(S, T) \leq \alpha \cdot dist_{\mathrm{sub}}(S', T)$ for all S-repairs S' of T, and an α-optimal U-repair is a U-repair U of T such that $dist_{\mathrm{upd}}(U, T) \leq \alpha \cdot dist_{\mathrm{upd}}(U', T)$ for all U-repairs U' of T. In particular, an optimal S-repair (respectively, optimal U-repair) is the same as a 1-optimal S-repair (respectively, 1-optimal U-repair).

Example 2.3. In our running example (Figure 1), tables S_1, S_2, and S_3 are consistent subsets, and U_1, U_2, and U_3 are consistent updates. For clarity, we marked with yellow shading the values that were changed for constructing each U_i. We have $dist_{\mathrm{sub}}(S_1, T) = 2$, since the missing tuple (tuple 1) has the weight 2. We also have $dist_{\mathrm{sub}}(S_2, T) = 2$ and $dist_{\mathrm{sub}}(S_3, T) = 3$. The reader can verify that S_1 and S_2 are optimal S-repairs. Table S_3 is *not* an optimal S-repair (and, in fact, not an S-repair at all), since the second tuple can be added to S_3 without violating consistency. Similarly, we have $dist_{\mathrm{upd}}(U_1, T) = 2$, $dist_{\mathrm{upd}}(U_2, T) = 3$, and $dist_{\mathrm{upd}}(U_3, T) = 4$ (since U_3 is obtained by changing two values of a tuple of weight 2). The table U_1 is an optimal U-repair, while U_2 and U_3 are not.

It should be noted that the values of an update U of a table T are not necessarily taken from the *active domain* (i.e., values that occur in T). An example is the value `F01` of table U_1 in Figure 1(e). This has implications on the complexity of computing optimal U-repairs. We discuss a restriction on the allowed update values in Section 6.

2.4 Complexity

We adopt the conventional measure of *data complexity*, where the schema $R(A_1, \ldots, A_k)$ and dependency set Δ are assumed to be *fixed*, and only the table T is considered the input. In particular, a "polynomial" running time may have an exponential dependency on k, as in $O(|T|^k)$. Note that in our complexity analysis, we do not take into account the number of attributes, since it is assumed to be fixed. Hence, each combination of $R(A_1, \ldots, A_k)$ and Δ defines a distinct problem of finding an optimal repair (of the relevant type), and different combinations may feature different computational complexities.

Finding an optimal repair is a special case of an *NP optimization problem* [18]. To prove hardness of approximation, we will characterize the complexity in terms of *APX-hardness* and *APX-completeness*, and the formal definitions are as follows.

An *optimization problem* is a tuple $(\mathcal{I}, Sol, \text{cost}, \text{goal})$ where $\mathcal{I}$ is a set of *instances* (i.e., input encodings), Sol maps every instance $x \in \mathcal{I}$ into a space $Sol(x)$ of *solutions*, cost is a function that maps every x and y into a number $\text{cost}(x, y) \in \mathbb{Q}$, and $\text{goal} \in \{\min, \max\}$ (i.e., the problem is either a minimization or a maximization problem, respectively). Given an instance x, the goal is to compute an optimal solution $\text{opt}(x) \in Sol(x)$ that maximizes or minimizes $\text{cost}(x, y)$ over all solutions $y \in Sol(x)$, depending on the type of the problem. We say that $(\mathcal{I}, Sol, \text{cost}, \text{goal})$ is an *NP optimization problem* if:

- the length of solutions $y \in Sol(x)$ is bounded by a polynomial in the length of x;
- both $x \in \mathcal{I}$ and $y \in Sol(x)$ can be decided in polynomial time;
- $\text{cost}(x, y)$ is computable in polynomial time, given $x \in \mathcal{I}$ and $y \in Sol(x)$.

For $\alpha \geq 1$, an *α-approximation* for an optimization problem $(\mathcal{I}, Sol, \text{cost}, \text{goal})$ is an algorithm that, for every instance x, produces an *α-optimal* solution y, that is, a solution $y \in Sol(x)$ such that $\max\{\frac{\text{cost}(x,y)}{\text{cost}(x,\text{opt}(x))}, \frac{\text{cost}(x,\text{opt}(x))}{\text{cost}(x,y)}\} \leq \alpha$. The complexity class APX is the class of all NP optimization problems that have a polynomial-time α-approximation for some constant $\alpha \geq 1$.

Let $P = (\mathcal{I}, Sol, \text{cost}, \text{goal})$ be an NP optimization problem. A *Polynomial-Time Approximation Scheme* (PTAS) for P is an algorithm A that takes as input an instance $x \in \mathcal{I}$ and $\alpha > 1$, and returns a solution $A(x, \alpha) \in Sol(x)$, such that for every $\alpha > 1$ it is the case that the algorithm $A(\cdot, \alpha)$ is a polynomial-time α-approximation for P [43].

Let $P = (\mathcal{I}, Sol, \text{cost}, \text{goal})$ and $P' = (\mathcal{I}', Sol', \text{cost}', \text{goal}')$ be NP optimization problems. A *PTAS reduction* from P to P' is a triple (f, g, κ) of algorithms that enable to transform a PTAS for P' into a PTAS for P:

- f transforms, in polynomial time, every instance $x \in \mathcal{I}$ into an instance $f(x) \in \mathcal{I}'$;
- κ maps every rational $\alpha > 1$ into a rational $c(\alpha) > 1$;
- g transforms, in polynomial time, every $\kappa(\alpha)$-optimal solution y' for $f(x)$, under P', into an α-optimal solution $g(x, y')$ for x, under P.

A *strict* reduction from P to P' is a pair (f, g) such that (f, g, κ) is a PTAS reduction where κ is the identity function; that is, $g(x, y')$ transforms an α-optimal solution for $f(x)$ into an α-optimal solution for x.

An NP optimization problem Q is *APX-hard* if there is a PTAS reduction from P to Q for every problem P in APX. As usual, Q is *APX-complete* if Q is in APX and Q is APX-hard. Throughout the article, we give several examples of known APX-complete problems, such as the *vertex-cover* minimization problem: find a smallest set of nodes that hits all edges of a given graph. It is known that an APX-hard problem does not have a PTAS, unless P = NP. Moreover, if Q is APX-hard, then there is a constant $\alpha > 1$ such that Q does not have any polynomial-time α-approximation, assuming P $\neq$ NP [30].

3 COMPUTING AN OPTIMAL S-REPAIR

In this section, we study the problem of computing an optimal S-repair. We begin with some notation and assumptions.

3.1 Setup

Throughout this section, we assume that every FD has a single attribute on its rhs, that is, it has the form $X \to A$. Clearly, this is not a limiting assumption, since replacing $X \to YZ$ with $X \to Y$ and $X \to Z$ preserves equivalence.

Let Δ be a set of FDs. If X is a set of attributes, then we denote by $\Delta - X$ the set Δ' of FDs that is obtained from Δ by removing each attribute of X from every lhs and rhs of every FD in Δ. Hence, no attribute in X occurs in $\Delta - X$. If A is an attribute, then we may write $\Delta - A$ instead of $\Delta - \{A\}$.

An *lhs marriage* of an FD set Δ is a pair (X_1, X_2) of distinct lhs of FDs in Δ such that:

- $cl_\Delta(X_1) = cl_\Delta(X_2)$;
- The lhs of every FD in Δ contains either X_1 or X_2 (or both).

Example 3.1. A simple example of an FD set with an lhs marriage is the following FD set:

$$\Delta_{A \leftrightarrow B \to C} \stackrel{\text{def}}{=} \{A \to B\,,\ B \to A\,,\ B \to C\}. \tag{1}$$

As another example, consider the following FD set:

$$\Delta_1 \stackrel{\text{def}}{=} \{\mathsf{ssn} \to \mathsf{first}\,,\ \mathsf{ssn} \to \mathsf{last}\,,\ \mathsf{first\ last} \to \mathsf{ssn}\,,\ \mathsf{ssn} \to \mathsf{address}\,,\ \mathsf{ssn\ office} \to \mathsf{phone}\,,\ \mathsf{ssn\ office} \to \mathsf{fax}\}.$$

Under Δ_1, the pair $(\{\mathsf{ssn}\}, \{\mathsf{first}, \mathsf{last}\})$ is an lhs marriage.

Finally, if S is a subset of a table T, then we denote by $w_T(S)$ the sum of weights of the tuples of S, that is,

$$w_T(S) \stackrel{\text{def}}{=} \sum_{i \in ids(S)} w_T(i)\,.$$

3.2 Algorithm

We now describe an algorithm for finding an optimal S-repair. The algorithm terminates in polynomial time, even under combined complexity, yet it may *fail*. If it succeeds, then the result is guaranteed to be an optimal S-repair. We later discuss the situations in which the algorithm fails. The algorithm, OptSRepair, is shown as Algorithm 1. The input is a set Δ of FDs and a table T, both over the same relation schema (that we do not need to refer to explicitly). In the remainder of this section, we fix Δ and T, and describe the execution of OptSRepair on Δ and T. In the pseudocode, we use conventional operators in relational algebra: projection (π), selection (σ) and union ($\cup$).

The algorithm handles four cases. The first is where Δ is trivial. Then, T is itself an optimal S-repair. The second case is where Δ has a common lhs A. Then, the algorithm groups the tuples by A, finds an optimal S-repair for each group (via a recursive call to OptSRepair), this time by ignoring A (i.e., removing A from the FDs of Δ), and returning the union of the optimal S-repairs. The precise description is in Subroutine 1 (CommonLHSRep). The third case is where Δ has a consensus FD $\emptyset \to A$. Similarly to the second case, the algorithm groups the tuples by A and finds an optimal S-repair for each group. This time, however, the algorithm returns an optimal S-repair with a maximal weight among these repairs. The precise description is in Subroutine 2 (ConsensusRep).

The fourth (last) case is the most involved. This is the case where Δ has an lhs marriage (X_1, X_2). In this case, the problem is reduced to finding a maximum weighted matching of a bipartite graph [34]. The maximum weight matching problem is the problem of finding, given a weighted bipartite graph (i.e., a bipartite graph in which every edge is associated with a weight), a matching in which the sum of the weights is maximal. The graph, which we denote by $G = (V_1, V_2, E, w)$, consists of two disjoint node sets V_1 and V_2, an edge set E that connects nodes from V_1 to nodes from V_2, and a weight function w that assigns a weight $w(v_1, v_2)$ to each edge (v_1, v_2). For $i = 1, 2$,

the node set V_i is the set of tuples in the projection of T to X_i.[1] To determine the weight $w(v_1, v_2)$, we select from T the subset T_{v_1,v_2} that consists of the tuples that agree with v_1 and v_2 on X_1 and X_2, respectively. We then find an optimal S-repair for T_{v_1,v_2}, after we remove from Δ every attribute in either X_1 or X_2. Then, the weight $w(v_1, v_2)$ is the weight of this optimal S-repair. Next, we find a maximum matching $E_{\max}$ of G. Note that $E_{\max}$ is a subset of E such that no node appears more than once. The returned result is then the disjoint union of the optimal S-repairs of T_{v_1,v_2} over all (v_1, v_2) in $E_{\max}$. The precise description is in Subroutine 3 (MarriageRep).

ALGORITHM 1: OptSRepair(Δ, T)

if Δ is trivial **then** ▷ *successful termination*
  return T
remove trivial FDs from Δ
if Δ has a common lhs **then**
  return CommonLHSRep(Δ, T)
if Δ has a consensus FD **then**
  return ConsensusRep(Δ, T)
if Δ has an lhs marriage **then**
  return MarriageRep(Δ, T)
fail ▷ *cannot find an optimal S-repair*

Subroutine 1: CommonLHSRep(Δ, T)

$A :=$ a common lhs of Δ
return $\cup_{(a)\in\pi_A T[*]}$OptSRepair($\Delta - A, \sigma_{A=a}T$)

Subroutine 2: ConsensusRep(Δ, T)

select a consensus FD $\emptyset \rightarrow A$ in Δ
for all $(a) \in \pi_A T[*]$ **do**
  $S_a :=$ OptSRepair($\Delta - A, \sigma_{A=a}T$)
$a_{\max} := \operatorname{argmax}_a\{w_T(S_a) \mid (a) \in \pi_A T[*]\}$
return $S_{a_{\max}}$

Subroutine 3: MarriageRep(Δ, T)

select an lhs marriage (X_1, X_2) of Δ
for all $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2}T[*]$ **do**
  $S_{\mathbf{a}_1,\mathbf{a}_2} :=$ OptSRepair($\Delta - X_1X_2, \sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$)
  $w(\mathbf{a}_1, \mathbf{a}_2) := w_T(S_{\mathbf{a}_1,\mathbf{a}_2})$
$V_i := \pi_{X_i}T[*]$ for $i = 1, 2$
$E := \{(\mathbf{a}_1, \mathbf{a}_2) \mid (\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2}T[*]\}$
$G :=$ weighted bipartite graph (V_1, V_2, E, w)
$E_{\max} :=$ a maximum matching of G
return $\cup_{(\mathbf{a}_1,\mathbf{a}_2)\in E_{\max}} S_{\mathbf{a}_1,\mathbf{a}_2}$

[1] In principle, it may be the case that the same tuple occurs in both V_1 and V_2, since the tuple is in both projections. Nevertheless, we still treat the two occurrences of the tuple as distinct nodes, and so effectively assume that V_1 and V_2 are disjoint.

The following example illustrates the execution of Subroutine 3. Example 3.5, given in the next subsection, illustrates the evolution of FD sets during the execution of OptSRepair.

Example 3.2. Let $\Delta = \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$ be a set of FDs over $R(A, B, C)$, and let T be a table that contains four tuples: $(1, 0, 0)$, $(1, 1, 0)$, $(0, 1, 1)$, $(0, 1, 2)$. Suppose that the weight of $(1, 1, 0)$ is 3, while the weight of the rest of the tuples is 1. The FD set Δ has an lhs marriage $(\{A\}, \{B\})$. There are three pairs of values in $\pi_{AB}T[*]$, namely $(1, 0)$, $(1, 1)$, and $(0, 1)$. In the subroutine MarriageRep, we first find an optimal S-repair for the table corresponding to each such pair. Clearly, an optimal S-repair of the table $\sigma_{A=1, B=0}T$ (that contains the tuple $(1, 0, 0)$) is the table itself, and $w(1, 0) = 1$. Similarly, the table $\sigma_{A=1, B=1}T$ (that contains the tuple $(1, 1, 0)$) is consistent and $w(1, 1) = 3$. Finally, for the table $\sigma_{A=0, B=1}T$ (that contains the tuples $(0, 1, 1)$, $(0, 1, 2)$), we have that $w(0, 1) = 1$, since these two tuples violate the FD $B \rightarrow C$ (hence only one of them will be in a repair).

The bipartite graph G will contain two nodes v_0, v_1 (corresponding to the values $0, 1$ in attribute A) on the left-hand-side, and two nodes u_0, u_1 (corresponding to the values $0, 1$ in attribute B) on the right-hand side. The edge set E will contain the edges (v_1, u_0) (with $w(v_1, u_0) = 1$), (v_1, u_1) (with $w(v_1, u_1) = 3$), and (v_0, u_1) (with $w(v_0, u_1) = 1$). A maximum weight matching of G will contain a single edge (v_1, u_1), and the corresponding table that contains the tuple $(1, 1, 0)$ is indeed an optimal S-repair of T. If, however, the weight of the tuple $(1, 1, 0)$ is also 1, then the weight of the edge (v_1, u_1) will be 1, and a maximum weight matching will contain two edges (v_1, u_0) and (v_0, u_1). In this case, there are two optimal S-repairs: one that contains both $(1, 0, 0)$ and $(0, 1, 1)$, and one that contains both $(1, 0, 0)$ and $(0, 1, 2)$. The algorithm will return one of these repairs.

The following theorem states the correctness and efficiency of OptSRepair.

Theorem 3.3. *Let Δ and T be a set of FDs and a table, respectively, over a relation schema $R(A_1, \ldots, A_k)$. If* OptSRepair(Δ, T) *succeeds, then it returns an optimal S-repair. Moreover,* OptSRepair(Δ, T) *terminates in polynomial time in k, $|\Delta|$, and $|T|$.*

In the next subsection, we discuss the cases where OptSRepair(Δ, T) fails. The complete proof of Theorem 3.3 is given in Section 4. The proof is by induction on the number of simplifications that OptSRepair applies to Δ. For each one of the three simplifications, we prove that if OptSRepair returns an optimal S-repair after the simplification is applied, then it also returns an optimal S-repair for the original set of FDs.

3.3 Dichotomy

The reader can observe that the success or failure of the algorithm OptSRepair(Δ, T) depends only on Δ, and not on T. The algorithm OSRSucceeds(Δ), depicted as Algorithm 2, tests whether Δ is such that OptSRepair succeeds by simulating the cases and corresponding changes to Δ. The next theorem shows that, under conventional complexity assumptions, OptSRepair covers *all* sets Δ such that an optimal S-repair can be found in polynomial time. Hence, we establish a dichotomy in the complexity of computing an optimal S-repair.

Theorem 3.4. *Let Δ be a set of FDs.*

- *If* OSRSucceeds(Δ) *returns true, then an optimal S-repair can be computed in polynomial time by executing* OptSRepair(Δ, T) *on the input T.*
- *If* OSRSucceeds(Δ) *returns false, then computing an optimal S-repair is APX-complete, and remains APX-complete on unweighted, duplicate-free tables.*

Moreover, the execution of OSRSucceeds(Δ) *terminates in polynomial time in $|\Delta|$.*

The proof of Theorem 3.4 is involved, hence we give the full proof in Section 4. Note that the positive side of the dichotomy is not guaranteed by traditional conditions on FDs, such as

ALGORITHM 2: OSRSucceeds(Δ)

```
1:  while Δ is nontrivial do
2:      remove trivial FDs from Δ
3:      if Δ has a common lhs A then
4:          Δ := Δ − A
5:      else if Δ has a consensus FD ∅ → A then
6:          Δ := Δ − A
7:      else if Δ has an lhs marriage (X1, X2) then
8:          Δ := Δ − X1X2
9:      else
10:         return false
11: return true
```

Boyce-Codd normal form (BCNF) [17]. For example, the FD set $\{A \rightarrow B, B \rightarrow A\}$ over $R(A, B)$ is in BCNF, but computing an optimal S-repair is APX-complete in this case.

Example 3.5. We now illustrate the application of Theorem 3.4 to several FD sets. Consider first the FD set Δ of our running example. The execution of OSRSucceeds(Δ) transforms Δ as follows:

$$\begin{aligned} &\{\text{facility} \rightarrow \text{city}\,,\ \text{facility room} \rightarrow \text{floor}\} \\ (\text{common lhs}) \Rrightarrow &\{\emptyset \rightarrow \text{city}\,,\ \text{room} \rightarrow \text{floor}\} \\ (\text{consensus}) \Rrightarrow &\{\text{room} \rightarrow \text{floor}\} \\ (\text{common lhs}) \Rrightarrow &\{\emptyset \rightarrow \text{floor}\} \\ (\text{consensus}) \Rrightarrow &\{\}. \end{aligned}$$

Hence, OSRSucceeds(Δ) is true, and hence, an optimal S-repair can be found in polynomial time.

Next, consider the FD set $\Delta_{A \leftrightarrow B \rightarrow C}$ from Example 3.1. The algorithm OSRSucceeds($\Delta_{A \leftrightarrow B \rightarrow C}$) executes as follows:

$$\begin{aligned} &\{A \rightarrow B, B \rightarrow A, B \rightarrow C\} \\ (\text{lhs marriage}) \Rrightarrow &\{\emptyset \rightarrow C\} \\ (\text{consensus}) \Rrightarrow &\{\}. \end{aligned}$$

Hence, this is again an example of an FD set on the tractable side of the dichotomy.

As the last positive example, we consider the FD set Δ_1 of Example 3.1:

$$\begin{aligned} &\{\text{ssn} \rightarrow \text{first}\,,\ \text{ssn} \rightarrow \text{last}\,,\ \text{first last} \rightarrow \text{ssn}\,,\ \text{ssn} \rightarrow \text{address}\,, \\ &\text{ssn office} \rightarrow \text{phone}\,,\ \text{ssn office} \rightarrow \text{fax}\} \\ (\text{lhs marriage}) \Rrightarrow &\{\emptyset \rightarrow \text{address}\,,\ \text{office} \rightarrow \text{phone}\,,\ \text{office} \rightarrow \text{fax}\} \\ (\text{consensus}) \Rrightarrow &\{\text{office} \rightarrow \text{phone}\,,\ \text{office} \rightarrow \text{fax}\} \\ (\text{common lhs}) \Rrightarrow &\{\emptyset \rightarrow \text{phone}\,,\ \emptyset \rightarrow \text{fax}\} \\ (\text{consensus}) \Rrightarrow &\{\}. \end{aligned}$$

However, for $\Delta = \{A \rightarrow B, B \rightarrow C\}$, none of the conditions of OSRSucceeds(Δ) is true, and therefore, the algorithm returns false. It thus follows from Theorem 3.4 that computing an optimal S-repair is APX-complete (even if all tuple weights are the same and there are no duplicate tuples). The same applies to $\Delta = \{A \rightarrow B, C \rightarrow D\}$.

As another example, the following corollary of Theorem 3.4 generalizes the tractability of our running example to general chain FD sets.

Corollary 3.6. *If Δ is a chain FD set, then an optimal S-repair is computable in polynomial time.*

Proof. The reader can easily verify that when Δ is a chain FD set, OSRSucceeds(Δ) will reduce it to emptiness by repeatedly removing consensus attributes and common-lhs, as done in our running example. □

3.4 Most Probable Database

In this section, we draw a connection to the *Most Probable Database problem* (MPD) [28]. A table in our setting can be viewed as a relation of a *tuple-independent database* [20] if each weight is in the interval $[0, 1]$. In that case, we view the weight as the probability of the corresponding tuple, and we call the table a *probabilistic table*. Such a table T represents a probability space over the subsets of T, where a subset is selected by considering each tuple $T[i]$ independently and *selecting* it with the probability $w_T(i)$, or equivalently, deleting it with the probability $1 - w_T(i)$. Hence, the probability of a subset S, denoted $\Pr_T(S)$, is given by

$$\Pr_T(S) \stackrel{\text{def}}{=} \left(\prod_{i \in ids(S)} w_T(i)\right) \times \left(\prod_{i \in ids(T) \setminus ids(S)} (1 - w_T(i))\right). \tag{2}$$

Given a constraint φ over the schema of T, MPD for φ is the problem of computing a subset S that satisfies φ, and has the maximal probability among all such subsets. Here, we consider the case where φ is a set Δ of FDs. Hence, MPD for Δ is the problem of computing

$$\operatorname*{argmax}_{S \subseteq T,\ S \models \Delta} \Pr_T(S).$$

Gribkoff, Van den Broeck, and Suciu [28] proved the following dichotomy for *unary* FDs, which are FDs of the form $A \rightarrow X$ having a single attribute on their lhs.

Theorem 3.7. *[28] Let Δ be a set of unary FDs over a relational schema. MPD for Δ is either solvable in polynomial time or NP-hard.*

The question of whether such a dichotomy holds for *general* (not necessarily *unary*) FDs has been left open. The following corollary of Theorem 3.4 fully resolves this question.

Theorem 3.8. *Let Δ be a set of FDs over a relational schema. If* OSRSucceeds(Δ) *is true, then MPD for Δ is solvable in polynomial time; otherwise, MPD is NP-hard, and has no polynomial-time (multiplicative) approximation within any subexponential factor, unless P = NP.*

Proof. We first show a reduction from MPD to the problem of computing an optimal S-repair. Let T be an input for MPD. By a *certain tuple*, we refer to a tuple identifier $i \in ids(T)$ such that $w_T(i) = 1$. We assume that the set of certain tuples satisfies Δ collectively, since otherwise the probability of any consistent subset is zero (and we can select, e.g., the empty subset as a most likely solution). We can then replace each probability 1 with a probability that is smaller than, yet close enough to 1, so that every consistent subset that excludes a certain fact is less likely than any subset that includes all certain facts. In addition, as observed by Gribkoff et al. [28], tuples with probability at most 0.5 can be eliminated, since we can always remove them from any (consistent) subset without reducing the probability. Hence, we assume that $0.5 < w_T(i) < 1$ for all $i \in ids(T)$.

From (2), we conclude the following:

$$\Pr_T(S) = \left(\prod_{i \in ids(S)} \frac{w_T(i)}{1 - w_T(i)}\right) \times \left(\prod_{i \in ids(T)} (1 - w_T(i))\right) \propto \left(\prod_{i \in ids(S)} \frac{w_T(i)}{1 - w_T(i)}\right). \tag{3}$$

Note that $p \propto q$ means that the two numbers differ by a multiplicative factor that is the same for all possible worlds. The reason for the proportionality ($\propto$) is that all consistent subsets share the same right factor of the first product. While the weight of a consistent subset is the sum of its tuple weights, the probability of a possible world is obtained by multiplying probabilities. Hence, a probability is translated into a weight by taking its logarithm. Thus, we construct a table T' that is the same as T, except that $w_{T'}(i) = \log(w_T(i)/(1 - w_T(i)))$ for all $i \in ids(T')$. Since we assume that $w_T(i) > 0.5$ for all $i \in ids(T)$, it holds that $w_{T'}(i)$ is strictly positive for all $i \in ids(T')$. Then, a most likely database of T is the same[2] as an optimal S-repair of T'.

For the "otherwise" part, we show a reduction from the problem of computing an optimal S-repair of an *unweighted* table to MPD. The reduction is straightforward: given T, we set the weight $w_T(i)$ of each tuple to 0.9 (or any fixed number greater than 0.5). From Equation (2) it follows that a consistent subset is most probable if and only if it has a maximal number of tuples. From Equation (3), we conclude that the probability of a subset S is proportional to $(0.9/0.1)^{|S|} = 9^{|S|}$. Therefore, the gap between the probability of an S-repair S and that of an optimal S-repair O is $9^{|O|-|S|}$. The APX-hardness of finding an optimal subset repair implies that, assuming P $\neq$ NP, there is a fixed $\alpha > 1$, such that we cannot guarantee any S with $|T| - |S| < \alpha(|T| - |O|)$, or equivalently, we cannot guarantee $|O| - |S| < (\alpha - 1)(|T| - |O|)$. Note, however, that $(\alpha - 1)(|T| - |O|)$ is $\Omega(|T|)$. Therefore, we cannot get an approximation better than the exponential $9^{\Omega(|T|)}$. □

Comment 3.9. *When considering unary FDs, there is a disagreement between our tractability condition (Algorithm 2) and that of Gribkoff et al. [28]. In particular, the FD set $\Delta_{A \leftrightarrow B \rightarrow C}$ defined in Equation (1) is classified as polynomial time in our dichotomy while NP-hard by Gribkoff et al. [28]. This is due to a gap in their proof of hardness.*[3]

3.5 Approximation

An easy observation is that the computation of an optimal subset is easily reducible to the *weighted vertex-cover* problem—given a graph G where nodes are assigned nonnegative weights, find a vertex cover (i.e., a set C of nodes that intersects with all edges) with a minimal sum of weights. Indeed, given a table T, we construct the graph G that has $ids(T)$ as the set of nodes, and an edge between every i and j such that $T[i]$ and $T[j]$ contradict one or more FDs in Δ. Given a vertex cover C for G, we obtain a consistent subset S by deleting from T every tuple with an identifier in C. Clearly, this reduction is strict. As weighted vertex cover is 2-approximable in polynomial time [8], we conclude the same for optimal subset repairing.

Proposition 3.10. *For all FD sets Δ, a 2-optimal S-repair can be computed in polynomial time.*

While Proposition 3.10 is straightforward, it is of practical importance as it limits the severity of the lower bounds we established in this section. Moreover, we will later show that the proposition has implications on the problem of approximating an optimal U-repair.

[2]We do not need to make an assumption of infinite precision to work with logarithms, since the algorithms we use for computing an optimal S-repair can replace addition and subtraction with multiplication and division, respectively.
[3]This has been established in a private communication with the authors of Reference [28].

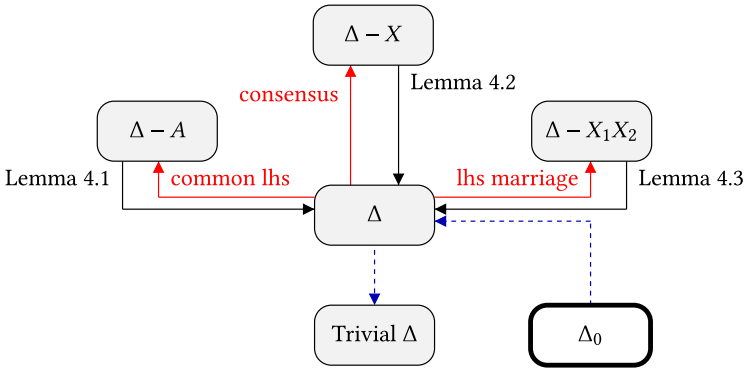

Fig. 2. An illustration of the proof of Theorem 3.3. We start with an FD set Δ_0 and apply simplifications to it, until we get a trivial set of FDs Δ. The red arrows represent simplifications. A black arrow from Δ' to Δ means that if we can find an optimal S-repair for Δ' in polynomial time, then we can find an optimal S-repair for Δ in polynomial time. The proof is in the lemma that appears next to the corresponding black arrow.

4 PROOF OF DICHOTOMY

In this section, we prove Theorem 3.4. As in the previous section, we assume that every FD has a single attribute on its rhs.

4.1 Positive Side

The positive side is a direct consequence of Theorem 3.3. Recall that Theorem 3.3 states that if OptSRepair(Δ, T) succeeds on a set of FDs Δ and a table T, then it return an optimal S-repair of T w.r.t. Δ in polynomial time. Since OSRSucceeds(Δ) simulates the execution of OptSRepair(Δ, T) and returns true if and only if OptSRepair(Δ, T) succeeds on every table T, the positive side of Theorem 3.4 follows immediately from Theorem 3.3. Hence, we start by proving this theorem.

The proof of Theorem 3.3 is illustrated in Figure 2. We prove the theorem by induction on the number of simplifications that OptSRepair applies to Δ. For each one of the three simplifications, we prove that if OptSRepair returns an optimal S-repair after the simplification is applied, then it also returns an optimal S-repair for the original set of FDs. We start by proving this for the common lhs simplification.

Lemma 4.1. *Let T be a table and Δ be a set of FDs that has a common lhs A. If* OptSRepair$(\Delta - A, \sigma_{A=a}T)$ *returns an optimal S-repair of $\sigma_{A=a}T$ w.r.t. $\Delta - A$ for all $(a) \in \pi_A T[*]$, then* CommonLHSRep(Δ, T) *returns an optimal S-repair of T w.r.t. Δ.*

Proof. Let J be the result of CommonLHSRep(Δ, T). We start by proving that J is consistent. Let us assume, by way of contradiction, that J is not consistent. Thus, there are two tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ in J that jointly violate an FD $Z \rightarrow B$ in Δ. Since A is a common lhs, it holds that $A \in Z$; hence, the tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ agree on the value of attribute A, and do not agree on the value of attribute B. Assume that $\mathbf{t}_1.A = \mathbf{t}_2.A = a$. By definition, there is an FD $(Z \setminus \{A\}) \rightarrow B$ in $\Delta - A$. Clearly, the tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ agree on all the attributes in $Z \setminus \{A\}$, and do not agree on the value of attribute B. Thus, $\mathbf{t}_1$ and $\mathbf{t}_2$ violate an FD in $\Delta - A$, which is a contradiction to the fact that OptSRepair$(\Delta - A, \sigma_{A=a}T)$ returns an optimal S-repair of $\sigma_{A=a}T$ that contains both $\mathbf{t}_1$ and $\mathbf{t}_2$.

Next, we prove that J is an optimal S-repair of T. Let us assume, by way of contradiction, that this is not the case. That is, there is another consistent subset J' of T, such that $w_T(J') > w_T(J)$. In this case, there exists at least one value a' of attribute A, such that the total weight of the tuples $\mathbf{t} \in J'$ for which it holds that $\mathbf{t}.A = a'$ is higher than the total weight of such tuples in J. Let

$F = \{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$ be the set of tuples from J for which it holds that $\mathbf{f}_j.A = a'$, and let $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_m\}$ be the set of such tuples in J'. It holds that $w_T(G) > w_T(F)$.

We claim that $\{\mathbf{g}_1, \ldots, \mathbf{g}_m\}$ is a consistent subset of $\sigma_{A=a'}T$. Let us assume, by way of contradiction, that $\{\mathbf{g}_1, \ldots, \mathbf{g}_m\}$ is not a consistent subset of $\sigma_{A=a'}T$. Thus, there exist two tuples $\mathbf{g}_{j_1}$ and $\mathbf{g}_{j_2}$ that jointly violate an FD, $Z \rightarrow B$, in $\Delta - A$. By definition, there is an FD $(Z \cup \{A\}) \rightarrow B$ in Δ, and since $\mathbf{g}_{j_1}$ and $\mathbf{g}_{j_2}$ agree on the value of attribute A, they clearly violate this FD as well, which is a contradiction to the fact that they both appear in J' (which is a consistent subset of T). Hence, G is a consistent subset of $\sigma_{A=a'}T$ and it holds that $w_T(G) > w_T(F)$, which is a contradiction to the fact that $\{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$ is an optimal S-repair of $\sigma_{A=a'}T$. We conclude that J is a consistent subset of T, and there is no other consistent subset of T with a weight higher that $w_T(J)$; hence J is an optimal S-repair of T w.r.t. Δ. □

Next, we consider the consensus FD simplification.

Lemma 4.2. *Let T be a table and Δ be a set of FDs that has a consensus FD $\emptyset \rightarrow A$. If* OptSRepair$(\Delta - A, \sigma_{A=a}T)$ *returns an optimal S-repair of $\sigma_{A=a}T$ w.r.t. $\Delta - A$ for all $(a) \in \pi_A T[*]$, then* ConsensusRep(Δ, T) *returns an optimal S-repair of T w.r.t. Δ.*

Proof. Let J be the result of ConsensusRep(Δ, T). We will start by proving that J is consistent. Let us assume, by way of contradiction, that J is inconsistent. Thus, there are two tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ in J that jointly violate an FD $Z \rightarrow B$ in Δ. Note that $\mathbf{t}_1$ and $\mathbf{t}_2$ agree on the value of attribute A (since ConsensusRep(Δ, T) always returns a set of tuples that agree on the value of attribute A). Assume that $\mathbf{t}_1.A = \mathbf{t}_2.A = a$. Therefore, it holds that $B \neq A$ and $\mathbf{t}_1.B \neq \mathbf{t}_2.B$. By definition, there is an FD $(Z \setminus \{A\}) \rightarrow B$ in $\Delta - A$. Clearly, the tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ agree on all the attributes in $Z \setminus \{A\}$, but do not agree on the value of attribute B. Thus, $\mathbf{t}_1$ and $\mathbf{t}_2$ also jointly violate an FD in $\Delta - A$, which is a contradiction to the fact that OptSRepair$(\Delta - A, \sigma_{A=a}T)$ returns an optimal S-repair of $\sigma_{A=a}T$ that contains both $\mathbf{t}_1$ and $\mathbf{t}_2$.

Next, we prove that J is an optimal S-repair of T. Clearly, an optimal S-repair of T is also an optimal S-repair of $\sigma_{A=a'}T$ for some $(a') \in \pi_A T[*]$ (as Δ contains the FD $\emptyset \rightarrow A$). We know that OptSRepair$(\Delta - A, \sigma_{A=a}T)$ returns an optimal S-repair of $\sigma_{A=a}T$ w.r.t. $\Delta - A$ for all $(a) \in \pi_A T[*]$, and ConsensusRep(Δ, T) chooses the S-repair with the highest weight among these S-repairs; hence, J is an optimal S-repair of T w.r.t. Δ. □

We now prove the above for the lhs-marriage simplification.

Lemma 4.3. *Let T be a table and Δ be a set of FDs that has an lhs marriage (X_1, X_2). If* OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T)$ *returns an optimal S-repair of $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$ w.r.t. $\Delta - X_1X_2$ for all $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2}T[*]$, then* MarriageRep$(\Delta, T)$ *returns an optimal S-repair of T w.r.t. Δ.*

Proof. Let J be the result of MarriageRep(Δ, T). We first prove that J is consistent. Assume, by way of contradiction, that $\mathbf{t}_1$ and $\mathbf{t}_2$ are two tuples in J that jointly violate Δ. We first observe that $\mathbf{t}_1[X_1] = \mathbf{t}_2[X_1]$ if and only if $\mathbf{t}_1[X_2] = \mathbf{t}_2[X_2]$, since J is constructed via a matching of G (lines 8-9). If $\mathbf{t}_1[X_1] \neq \mathbf{t}_2[X_1]$ and $\mathbf{t}_1[X_2] \neq \mathbf{t}_2[X_2]$, then the definition of an lhs marriage implies that $\mathbf{t}_1$ and $\mathbf{t}_2$ disagree on the left-hand side of every FD in Δ, and hence satisfy Δ. We conclude that $\mathbf{t}_1[X_1] = \mathbf{t}_2[X_1]$ and $\mathbf{t}_1[X_2] = \mathbf{t}_2[X_2]$, and therefore, $\mathbf{t}_1$ and $\mathbf{t}_2$ are both in $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$ for some $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2}T[*]$. Suppose that $\mathbf{t}_1$ and $\mathbf{t}_2$ violate the FD $Z \rightarrow B$ in Δ. Since $\mathbf{t}_1$ and $\mathbf{t}_2$ agree on X_1 and X_2, the tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ must violate $Z \setminus (X_1 \cup X_2) \rightarrow B$, which is in $\Delta - X_1X_2$. This contradicts the assumption that OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=a_1, X_2=a_2}T)$ returns an S-repair. We conclude that J is consistent, as claimed.

We complete the proof by showing that J is optimal. Let J' be a consistent subset of T. We need to prove that $w_T(J') \leq w_T(J)$. From the construction of J it follows that $w_T(J) = w(E_{\max})$, where

$w(E)$ denotes the sum of weights of a matching E of G. So, it suffices to prove that $w_T(J') \leq w(E')$ for some matching E' of G.

The definition of an lhs marriage implies that both $X_1 \rightarrow X_2$ and $X_2 \rightarrow X_1$ are entailed by Δ. Hence, if $\mathbf{t}_1$ and $\mathbf{t}_2$ are tuples of J', then it is again the case that $\mathbf{t}_1[X_1] = \mathbf{t}_2[X_1]$ if and only if $\mathbf{t}_1[X_2] = \mathbf{t}_2[X_2]$. We select as E' the matching of G that contains the edges $(\mathbf{a}_1, \mathbf{a}_2)$ whenever $\mathbf{t}[X_1] = \mathbf{a}_1$ and $\mathbf{t}[X_2] = \mathbf{a}_2$ for some tuple $\mathbf{t}$ of J'. As J' is consistent, we have that $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2} J'$ is a consistent subset of $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2} T$ w.r.t. $\Delta - X_1X_2$ for all $(\mathbf{a}_1, \mathbf{a}_2) \in E'$, since all the tuples in this instance agree on X_1X_2. Then, if $S_{\mathbf{a}_1,\mathbf{a}_2}$ is an optimal S-repair of $\sigma_{X_1=a_1, X_2=a_2} T$, then $w_T(\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2} J') \leq w_T(S_{\mathbf{a}_1,\mathbf{a}_2})$. Moreover, $w_T(S_{\mathbf{a}_1,\mathbf{a}_2}) = w(\mathbf{a}_1, \mathbf{a}_2)$ due to the construction of G and the assumption that OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=a_1, X_2=a_2} T)$ returns an optimal S-repair. Thus, $w_T(J') \leq w(E')$ as claimed. □

Finally, we use the above lemmas to prove Theorem 3.3. We will prove the theorem by induction on n, the number of simplifications that will be applied to Δ by OptSRepair. We start by proving the basis of the induction, that is, $n = 0$. In this case, OptSRepair(Δ, T) will only succeed if $\Delta = \emptyset$ or if Δ is trivial. Clearly, in this case, T is consistent w.r.t. Δ and an optimal S-repair of T is T itself. And indeed, OptSRepair(Δ, T) will return T.

For the inductive step, we need to prove that if the claim is true for all $n = 1, \ldots, k-1$, it is also true for $n = k$. In this case, OptSRepair(Δ, T) will start by applying some simplification to the schema. Clearly, the result is a set of FDs Δ', such that OptSRepair(Δ', T') will apply $k - 1$ simplifications to Δ'. One of the following holds:

(1) Δ has a common lhs A. In this case, the condition of line 4 is satisfied and the subroutine CommonLHSRep will be called. Note that OptSRepair(Δ, T) will succeed if and only if OptSRepair$(\Delta - A, \sigma_{A=a} T)$ succeeds for each $(a) \in \pi_A T[*]$. We know from the inductive assumption that if OptSRepair$(\Delta - A, \sigma_{A=a} T)$ succeeds, then it returns] an optimal S-repair. Thus, Lemma 4.1 implies that OptSRepair(Δ, T) returns an optimal S-repair of T w.r.t. Δ.

(2) Δ does not have a common lhs, but has a consensus FD $\emptyset \rightarrow A$. In this case, the condition of line 4 is not satisfied, but the condition of line 6 is satisfied and the subroutine ConsensusRep will be called. Again, OptSRepair(Δ, T) will succeed if and only if OptSRepair$(\Delta - A, \sigma_{A=a} T)$ succeeds for each $(a) \in \pi_A T[*]$. We know from the inductive assumption that if OptSRepair$(\Delta - A, \sigma_{A=a} T)$ succeeds, then it returns an optimal S-repair. Thus, Lemma 4.2 implies that OptSRepair(Δ, T) returns an optimal S-repair of T w.r.t. Δ.

(3) Δ does not have a common lhs or a consensus FD, but has an lhs marriage. In this case, the conditions of line 4 and line 6 are not satisfied, but the condition of line 8 is satisfied and the subroutine MarriageRep will be called. As in the previous cases, OptSRepair(Δ, T) will succeed if and only if OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2} T)$ succeeds for each $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2} T[*]$. We know from the inductive assumption that if OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2} T)$ succeeds, then it returns an optimal S-repair. Thus, Lemma 4.3 implies that OptSRepair(Δ, T) returns an optimal S-repair of T w.r.t. Δ.

This concludes our proof of correctness of algorithm OptSRepair.

4.1.1 Complexity. Next, we prove the complexity claim of Theorem 3.3. The main observation here is that whenever the algorithm makes a recursive call, it is applied to disjoint sets of tuples of T. For the common lhs and consensus FD simplifications, the recursive call is applied to $\sigma_{A=a} T$ for every $(a) \in \pi_A T[*]$, and for the lhs marriage simplification, the recursive call is applied to $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2} T$ for every $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2} T[*]$.

We now provide the recurrence function for each one of the subroutines of the algorithm. The following is the recurrence function for the subroutines CommonLHSRep and ConsensusRep:

$$F(k,n) \;\leq\; P(k,n) + \sum_{(a)\in\pi_A T[*]} F(k-1, n_a),$$

where k is the number of attributes that occur in Δ, n is the number of tuples in T, P is a polynomial, and n_a is the number of tuples in $\sigma_{A=a}T$. These subroutines have the same recurrence function (up to the polynomial P), since in both cases we split the database into block of tuples that agree on the value of attribute A, and then solve the problem separately for each one of these blocks.

The recurrence function for the subroutine MarriageRep looks as follows:

$$F(k,n) \;\leq\; P(k,n) + \sum_{\substack{(\mathbf{a}_1,\,\mathbf{a}_2)\,\in\\ \pi_{X_1X_2}T[*]}} F(k-1, n_{\mathbf{a}_1,\mathbf{a}_2}),$$

where $n_{\mathbf{a}_1,\mathbf{a}_2}$ is the number of tuples in $\sigma_{X_1=\mathbf{a}_1,X_2=\mathbf{a}_2}T$.

Since finding an optimal S-repair for a trivial FD set requires no computation (except for returning the table itself), and since in each one of the above recurrence functions the tables in the last argument form a partition of T, a standard analysis of F shows that it is bounded by a polynomial.

4.2 Negative Side

For the negative side of Theorem 3.4, membership in APX is due to Proposition 3.10. The proof of hardness is based on the concept of a *fact-wise reduction* [31], as previously done for proving dichotomies on sets of FDs [23, 31, 32, 36]. In our setup, a fact-wise reduction is defined as follows. Let R and R' be two relation schemas. A *tuple mapping* from R to R' is a function μ that maps tuples over R to tuples over R'. We extend μ to map tables T over R to tables $\mu(T)$ over R' in the following way. The table $\mu(T)$ will have the same tuple identifiers as T (i.e., $ids(\mu(T)) = ids(T)$). Moreover, $\mu(T)[i] = \mu(T[i])$ and $w_{\mu(T)}(i) = w_T(i)$ for all $i \in ids(\mu(T))$. Let Δ and Δ' be sets of FDs over R and R', respectively. A *fact-wise reduction* from (R,Δ) to (R',Δ') is a tuple mapping Π from R to R' with the following properties:

(1) Π is injective; that is, for all tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ over R, if $\Pi(\mathbf{t}_1) = \Pi(\mathbf{t}_2)$ then $\mathbf{t}_1 = \mathbf{t}_2$;
(2) Π preserves consistency and inconsistency; that is, for all tables T, the table $\Pi(T)$ satisfies Δ' if and only if T satisfies Δ;
(3) Π is computable in polynomial time.

The following lemma is straightforward.

Lemma 4.4. *Let R and R' be relation schemas and Δ and Δ' FD sets over R and R', respectively. If there is a fact-wise reduction from (R,Δ) to (R',Δ'), then there is a strict reduction from the problem of computing an optimal S-repair under R and Δ to that of computing an optimal S-repair under R' and Δ'.*

Our proof, which is illustrated in Figure 3, consists of four steps.

(1) We first prove APX-hardness for each of the FD sets in Table 1 over $R(A,B,C)$. For $\Delta_{A\to B\to C}$ and $\Delta_{A\to C\leftarrow B}$, we adapt reductions by Gribkoff et al. [28] in the work that we discussed in Section 3.4. For $\Delta_{AB\to C\to B}$, we show a reduction from MAX-NM-SAT [29]. Most intricate is the proof for $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$, where we devise a nontrivial adaptation of a reduction by Amini et al. [3] from finding a maximum bounded covering by 3-sets to triangle packing in graphs of bounded degree.

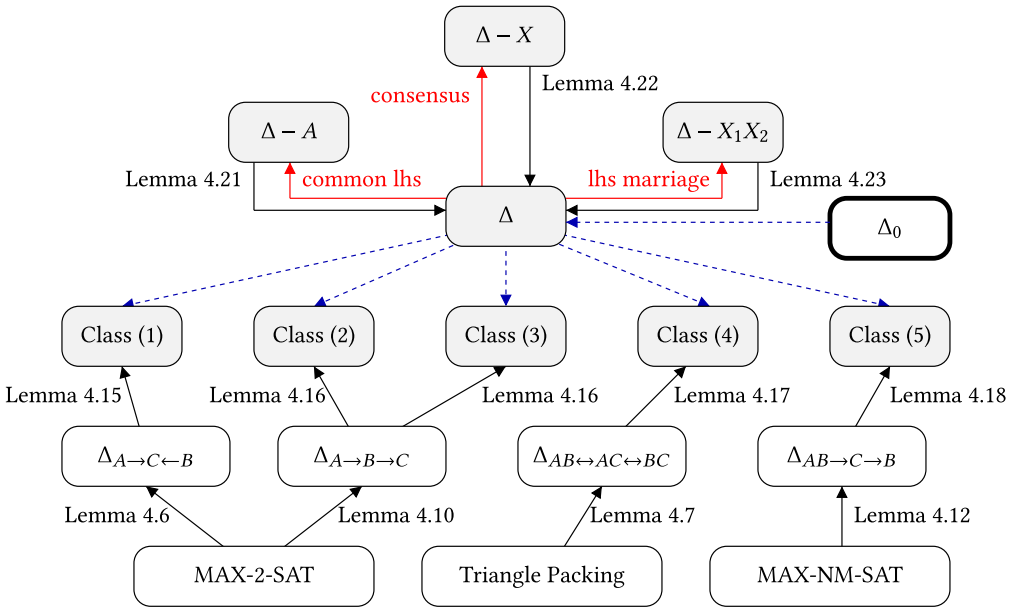

Fig. 3. An illustration of our proof of the negative side of Theorem 3.4. We start with an FD set Δ_0 and apply simplifications to it, until we get a nontrivial set of FDs Δ, that we classify into one of five classes. The red arrows represent simplifications. A black arrow represents a reduction that we construct in the lemma that appears next to the arrow.

Table 1. FD Sets Over $R(A, B, C)$ Used in the Proof of Hardness of Theorem 3.4

Name	FDs
$\Delta_{A\to B\to C}$	$A \to B, B \to C$
$\Delta_{A\to C\leftarrow B}$	$A \to C, B \to C$
$\Delta_{AB\to C\to B}$	$AB \to C, C \to B$
$\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$	$AB \to C, AC \to B, BC \to A$

(2) Next, we consider an FD set Δ that cannot be further simplified (that is, Δ does not have a common lhs, a consensus FD, or an lhs marriage). We show that Δ can be classified into one of five certain classes of FD sets.
(3) Then, we prove that for each FD set Δ in one of the five classes there exists a fact-wise reduction from one of the four schemas of Table 1.
(4) Finally, we prove that whenever OSRSucceeds simplifies Δ into Δ', there is a fact-wise reduction from (R, Δ') to (R, Δ), where R is the underlying relation schema.

Next, we give the full proof. Recall that in the negative side of Theorem 3.4, we consider unweighted, duplicate-free tables; hence, in all of our reductions, we will construct such tables. Also recall that to construct an α-optimal repair, it suffices to construct a consistent subset S of T such that $dist_{\text{sub}}(S, T) \leq \alpha \cdot dist_{\text{sub}}(S_{\text{opt}}, T)$, where S_{opt} is an optimal S-repair of T. We can then transform the consistent subset into a repair, without increasing the distance, by greedily adding tuples

while we can. Hence, in our proofs, we will often construct consistent subsets without reasoning about maximality.

4.2.1 Step 1: Hardness for the FD Sets in Table 1. We start by proving that computing an optimal S-repair for the FD sets in Table 1 is APX-complete. Proposition 3.10 implies that the problem is in APX for each one of these sets. Thus, it is only left to show that the problem is also APX-hard. Gribkoff et al. [28] proved that the MPD problem is NP-hard for both $\Delta_{A\to B\to C}$ and $\Delta_{A\to C\leftarrow B}$. Their hardness proof also holds for the problem of computing an optimal S-repair. Thus, we have the following lemma.

Lemma 4.5. *[28] Computing an optimal S-repair is NP-hard for both $\Delta_{A\to C\leftarrow B}$ and $\Delta_{A\to B\to C}$.*

We will now strengthen the above result by showing that the problems are not only NP-hard but also APX-hard. Gribkoff et al. [28] prove Lemma 4.5 by showing a reduction from the MAX-2-SAT problem: given a 2-CNF formula ψ, determine what is the maximum number of clauses in ψ, which can be simultaneously satisfied. We now show that their reduction is a PTAS-reduction from MAX-2-SAT to our problem, and since MAX-2-SAT is known to be APX-complete [7], we will conclude that our problem is APX-complete as well. We give the results separately for the convenience of later reference.

Lemma 4.6. *Computing an optimal S-repair for $\Delta_{A\to C\leftarrow B}$ is APX-complete.*

Proof. The reduction of Gribkoff et al. [28] uses the following construction. Given a 2-CNF formula ψ, they construct a table T over $R(A, B, C)$, by adding the tuples (c_i, x_k, l_{x_k}) and (c_i, x_j, l_{x_j}) for each clause $c_i = l_{x_j} \vee l_{x_k}$ in ψ, where l_x is either x or $\neg x$ for $x \in \{x_j, x_k\}$. The FD $A \to C$ in $\Delta_{A\to C\leftarrow B}$ ensures that each consistent subset contains at most one tuple for each clause. The FD $B \to C$ in $\Delta_{A\to C\leftarrow B}$ ensures that for every variable x, each consistent subset contains either x or $\neg x$ (but not both) in attribute C. Hence, as proved by Gribkoff et al. [28], it holds that the maximum number of clauses that can be simultaneously satisfied is exactly the size of an optimal S-repair of the constructed table T (which is unweighted and duplicate-free). We denote by n the number of clauses in ψ. Clearly, the constructed T contains $2n$ tuples. Hence, an optimal solution for MAX-2-SAT satisfies m clauses if and only if an optimal S-repair is obtained by deleting $2n - m$ tuples from T.

We now prove that their reduction is a PTAS-reduction (f, g, κ). The function f is described above. Clearly, it transforms an input x to MAX-2-SAT to an input $f(x)$ to our problem in polynomial time. Given a solution y' to $f(x)$, the solution $g(x, y')$ for x will assign the value 1 to a variable x_j in the formula if a tuple (c_i, x_j, x_j) is in $T \setminus y'$ (i.e., it belongs to the S-repair obtained by deleting the tuples in y' from T). Similarly, it will assign the value 0 to a variable x_k if a tuple $(c_i, x_k, \neg x_k)$ is in $T \setminus y'$. As explained above, a consistent subset of T cannot contain two tuples (c_i, x_j, x_j) and $(c_r, x_j, \neg x_j)$; hence, this is indeed a truth assignment to the variables in the formula. Moreover, the assignment $g(x, y')$ satisfies each c_i that appears in attribute A in some tuple from $T \setminus y'$, and since each tuple in $T \setminus y'$ has a different c_i in attribute A (otherwise, the FD $A \to C$ is violated), exactly $2n - |y'|$ clauses are satisfied by $g(x, y')$. Note that since $T \setminus y'$ is an S-repair, we cannot add tuples to it without violating consistency; hence, clauses c_i that do not appear in $T \setminus y'$ are not satisfied by this assignment. We will prove that if y' is a $\kappa(\alpha)$-optimal solution to our problem on $f(x)$, then $g(x, y')$ is an α-optimal solution to MAX-2-SAT on x, where $\kappa(\alpha) = \frac{4\alpha-1}{3\alpha}$.

Let y_{opt} be a set of clauses satisfied by an optimal solution to the MAX-2-SAT problem, and let y'_{opt} be an optimal solution to our problem. It is known that there is always an assignment that satisfies at least half of the clauses in the formula; hence, it holds that $\frac{1}{2}n \le |y_{\text{opt}}|$. Now, for $\beta > 1$, let y' be a β-optimal solution to our problem. Then, we have that $|y'| \le \beta|y'_{\text{opt}}|$. We will now show

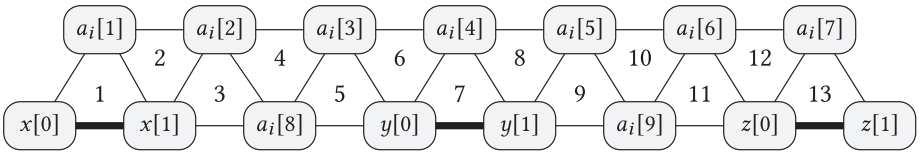

Fig. 4. An illustration of the reduction used by Amini et al. [3] to prove APX-hardness for the problem of finding the maximum number of edge disjoint triangles in a tripartite graph with a bounded degree.

that if $\beta = \frac{4\alpha-1}{3\alpha}$, then $g(x, y')$ is an α-optimal solution for MAX-2-SAT. Let y be the set of clauses satisfied by the assignment $g(x, y')$:

$$|y| = 2n - |y'| \geq 2n - \beta|y'_{\text{opt}}| = 2n - \beta(2n - |y_{\text{opt}}|) = \beta|y_{\text{opt}}| - 2(\beta - 1)n$$
$$\geq \beta|y_{\text{opt}}| - 2(\beta - 1) \cdot 2|y_{\text{opt}}| = (4 - 3\beta)|y_{\text{opt}}|.$$

Hence, the following holds:

$$\frac{|y|}{|y_{\text{opt}}|} \geq (4 - 3\beta) = \frac{1}{\alpha}$$
$$\Rightarrow \beta = \frac{4\alpha - 1}{3\alpha}.$$

Therefore, to obtain an α-optimal solution to MAX-2-SAT, we need to find a $\frac{4\alpha-1}{3\alpha}$-optimal solution to our problem. Clearly, $\kappa(\alpha) > 1$ whenever $\alpha > 1$, and that concludes our proof. □

Lemma 4.7. *Computing an optimal S-repair for $\Delta_{A\to B\to C}$ is APX-complete.*

Proof. Gribkoff et al. [28] use the same reduction for both $\Delta_{A\to C\leftarrow B}$ and $\Delta_{A\to B\to C}$. The only difference is that for $\Delta_{A\to B\to C}$ the FD $A \to B$ (rather than $A \to C$) ensures that each consistent subset contains at most one tuple for each clause. Hence, the proof of the previous lemma also holds for this case. □

Next, we prove that computing an optimal S-repair for $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$ is APX-complete as well. To do that, we consider the problem of finding the maximum number of edge-disjoint triangles in a tripartite graph with a bounded degree B. Amini et al. [3] (who refer to this problem as MECT-B) proved that this problem is APX-complete. We first prove that the complement problem of MECT-B (i.e., what is the minimum number of triangles that need to be left out in any choice of a set of edge-disjoint triangles from the graph) is APX-hard for tripartite graphs that satisfy a specific property. We denote the complement problem by $\overline{\text{MECT-B}}$. We use the reduction of Amini et al. [3] to prove that.

To prove APX-hardness for the problem MECT-B, they build a reduction from the problem of finding a maximum bounded covering by 3-sets: given a collection of subsets of a given set that contain exactly three elements each, such that each element appears in at most B subsets, find the maximum number of disjoint subsets. In their reduction, they construct a tripartite graph, such that for each subset $S_i = \{x, y, z\}$, they add to the graph the structure from Figure 4. Note that the nodes $a_i[1] \ldots a_i[9]$ are unique for this subset, while the nodes $x[0], x[1], y[0], y[1], z[0], z[1]$ will appear only once in the graph, even if they appear in more than one subset. Thus, we can build a set of edge-disjoint triangles for the constructed tripartite graph by selecting, for each subset, six out of the thirteen triangles (the even ones). This is true, since the even triangles do not share an edge with any other triangle. Hence, in their reduction, they construct a tripartite graph with the following property: the maximum number of edge-disjoint triangles in the graph is at least $\frac{6}{13}$ of the total number of triangles. We denote a graph that satisfies this property by $\frac{6}{13}$-tripartite graph. Thus, the reduction of Amini et al. [3] implies that the following holds.

Lemma 4.8. *The problem MECT-B for $\frac{6}{13}$-tripartite graphs is APX-hard.*

We will now prove that the complement problem is APX-hard as well.

Lemma 4.9. *The problem $\overline{\text{MECT-B}}$ on $\frac{6}{13}$-tripartite graphs is APX-hard.*

Proof. We construct a PTAS reduction (f, g, κ) from MECT-B on $\frac{6}{13}$-tripartite graphs to $\overline{\text{MECT-B}}$ on $\frac{6}{13}$-tripartite graphs. The function f is the identity function; that is, $f(x) = x$. Given a solution y' to $\overline{\text{MECT-B}}$, the solution $g(x, y')$ will contain every triangle that belongs to x, but does not belong to y'. We will now prove that if y' is an $\frac{13\alpha-6}{7\alpha}$-optimal solution for $\overline{\text{MECT-B}}$ on $f(x)$, then $g(x, y')$ is an α-optimal solution for MECT-B on x.

Let y_{opt} be an optimal solution to MECT-B on x, and let y'_{opt} be an optimal solution to $\overline{\text{MECT-B}}$ on $f(x)$. Let n be the number of triangles in x. Since the input to both problems is a $\frac{6}{13}$-tripartite graph, it holds that $|y_{\text{opt}}| \geq \frac{6}{13} \cdot n$. Now, for $\beta > 1$, let y' be a β-optimal solution to $\overline{\text{MECT-B}}$ on $f(x)$. Then, we have that $|y'| \leq \beta |y'_{\text{opt}}|$. We will show that if $\beta = \frac{13\alpha-6}{7\alpha}$, then $g(x, y')$ is an α-optimal solution for MECT-B on x. We denote by y the set $g(x, y')$ of edge-disjoint triangles. Note that if $|y'| = m$, then $|y| = n - m$. Moreover, if $|y'_{\text{opt}}| = m$, then $|y_{\text{opt}}| = n - m$:

$$|y| = n - |y'| \geq n - \beta |y'_{\text{opt}}| = n - \beta(n - |y_{\text{opt}}|) = \beta |y_{\text{opt}}| - (\beta - 1)n$$
$$\geq \beta |y_{\text{opt}}| - (\beta - 1) \cdot \frac{13}{6} |y_{\text{opt}}| = \left(\frac{13}{6} - \frac{7}{6}\beta\right) |y_{\text{opt}}|.$$

Hence, the following holds:

$$\frac{|y|}{|y_{\text{opt}}|} \geq \left(\frac{13}{6} - \frac{7}{6}\beta\right) = \frac{1}{\alpha}$$
$$\Rightarrow \beta = \frac{13\alpha - 6}{7\alpha}.$$

Thus, to obtain an α-optimal solution to MECT-B, we need to find a $\frac{13\alpha-6}{7\alpha}$-optimal solution to $\overline{\text{MECT-B}}$, and that concludes our proof. □

Next, we introduce our reduction from $\overline{\text{MECT-B}}$ on $\frac{6}{13}$-tripartite graphs to the problem of computing an optimal S-repair for $\Delta_{AB \leftrightarrow AC \leftrightarrow BC}$.

Lemma 4.10. *Computing an optimal S-repair for $\Delta_{AB \leftrightarrow AC \leftrightarrow BC}$ is APX-complete.*

Proof. We construct a strict reduction (f, g) from $\overline{\text{MECT-B}}$ on $\frac{6}{13}$-tripartite graphs. In our construction, tuples (a, b, c) will represent triangles (consisting of the nodes a, b and c), and the FDs will assert the edge disjointness. More formally, the input x to the $\overline{\text{MECT-B}}$ problem is a $\frac{6}{13}$-tripartite graph with a bounded degree B. We assume that x contains three sets of nodes: $\{a_1, \ldots, a_n\}$, $\{b_1, \ldots, b_l\}$ and $\{c_1, \ldots, c_r\}$. Given such an input, the function f will construct an input T for our problem as follows. For each triangle in x that consists of the nodes a_i, b_j, and c_k, the table T will contain a tuple (a_i, b_j, c_k). Given a solution y' to our problem on $f(x)$, the solution $g(x, y')$ will contain every triangle (a_i, b_j, c_k) corresponding to a tuple in y'. We will now prove that at most m triangles need to be left out in any choice of a set of edge-disjoint triangles from x if and only if there is a consistent subset of T that is obtained by deleting at most m tuples.

The "if" direction. Assume that there is a consistent subset J of T that is obtained by deleting at most m tuples. The FD $AB \rightarrow C$ implies that a consistent subset cannot contain two tuples (a_i, b_j, c_{k_1}) and (a_i, b_j, c_{k_2}) such that $c_{k_1} \neq c_{k_2}$. Moreover, the FD $AC \rightarrow B$ implies that it cannot contain two tuples (a_i, b_{j_1}, c_k) and (a_i, b_{j_2}, c_k) such that $b_{j_1} \neq b_{j_2}$, and the FD $BC \rightarrow A$ implies that it cannot contain two tuples (a_{i_1}, b_j, c_k) and (a_{i_2}, b_j, c_k) such that $a_{i_1} \neq a_{i_2}$. Thus, two triangles

$(a_{i_1}, b_{j_1}, c_{k_1})$ and $(a_{i_2}, b_{j_2}, c_{k_2})$ in x that correspond to two tuples $(a_{i_1}, b_{j_1}, c_{k_1})$ and $(a_{i_1}, b_{j_1}, c_{k_1})$ in J, will not share an edge (they can only share a single node). Hence, there is a set of edge-disjoint triangles from x that does not contain at most m triangles from x (one for each tuple in $T \setminus J$).

The "only if" direction. Assume that there is a set S of edge-disjoint triangles from g, such that at most m triangles from g do not belong to S. We can build a consistent subset J of T in the following way: for each triangle (a_i, b_j, c_k) in S, we will add the tuple (a_i, b_j, c_k) to J. Thus, J will contain $|S|$ tuples; that is, there will be at most m tuples in $T \setminus J$. It is only left to show that J is consistent. Let us assume, by way of contradiction, that J is not consistent. That is, there are two tuples (a_1, b_1, c_1) and (a_2, b_2, c_2) in J that violate an FD in $\Delta_{AB \leftrightarrow AC \leftrightarrow BC}$. If the tuples violate the FD $AB \rightarrow C$, then it holds that $a_1 = a_2$ and $b_1 = b_2$. Thus, the corresponding two triangles from S share the edge (a_1, b_1), which is a contradiction to the fact that S is a set of edge-disjoint triangles. Similarly, if the tuples violate the FD $AC \rightarrow B$, then the corresponding two triangles share an edge (a_1, c_1), and if they violate the FD $BC \rightarrow A$, the corresponding two triangles share an edge (b_1, c_1). Therefore, there exists a consistent subset of T that is obtained by deleteing at most m tuples.

The above implies that if y' is an α-optimal solution to our problem on $f(x)$, then $g(x, y')$ is an α-optimal solution to $\overline{\text{MECT-B}}$ on x. Hence, our reduction is a strict reduction from $\overline{\text{MECT-B}}$, and Lemma 4.9 implies that our problem is indeed APX-complete. □

Finally, we construct a reduction from MAX-NM-SAT to the problem of computing an optimal S-repair for $\Delta_{AB \rightarrow C \rightarrow B}$. MAX-NM-SAT is the problem of determining what is the maximum number of clauses that can be simultaneously satisfied in a CNF-formula where each clause contains either only positive literals or only negative literals. We start by proving that MAX-NM-SAT is APX-hard.

Lemma 4.11. *The problem MAX-NM-SAT is APX-hard.*

Proof. Guruswami [29] constructed a reduction from the MAX-3-SAT problem (which is the same as MAX-2-SAT, except that the input is a 3CNF formula) to the MAX-NM-SAT problem. Given an input x to the first problem (i.e., a 3CNF formula), he constructs an input $f(x)$ to the second problem by replacing every caluse c_i in x with two clauses: $c_i^p = (\bigvee_{l \in P_{c_i}} l \vee z_{c_i})$ and $c_i^n = (\bigvee_{l \in N_{c_i}} l \vee \neg z_{c_i})$, where P_{c_i} is the set of positive literals in c_i, N_{c_i} is the set of negative literals in c_i, and z_{c_i} is a new variable. We now prove that this reduction is a PTAS-reduction, and since MAX-3-SAT is APX-complete [43], this will conclude our proof.

Clearly, the function f described above transforms an input x to MAX-3-SAT to an input $f(x)$ to MAX-NM-SAT in polynomial time. The function g will transform a solution y' to MAX-NM-SAT on $f(x)$ to a solution y to MAX-3-SAT on x by projecting the assignment y' to the variables in x. That is, for every variable z that appears in x, we will have that $y(z) = 1$ if $y'(z) = 1$, and $y(z) = 0$ if $y'(z) = 0$. We now prove that if y' is a $\kappa(\alpha)$-optimal solution to MAX-NM-SAT on $f(x)$, then $g(x, y')$ is an α-optimal solution to MAX-3-SAT on x, where $\kappa(\alpha) = \frac{3\alpha}{2\alpha+1}$.

Let n be the number of clauses in x. Then, the number of clauses in $f(x)$ is $2n$. We start by showing that at least k clauses can be simultaneously satisfied in x if and only if at least $n + k$ clauses can be simultaneously satisfied in $f(x)$. Each clause c_i in x corresponds to two clauses c_i^p and c_i^n in $f(x)$. If c_i is satisfied by a positive literal, then c_i^p is satisfied by the same assignment, and we can satisfy c_i^n by assigning the value 0 to the variable z_{c_i}. A similar argument holds for the case where c_i is satisfied by a negative literal, in which case, we will assign the value 1 to z_{c_i}. If c_i is not satisfied, then by assigning the value 1 to z_{c_1}, we can satisfy the clause c_i^p, but the clause c_i^n will not be satisfied. Hence, an assignment that satisfies k clauses in x can be extended to an assignment that satisfies $n + k$ clauses in $f(x)$. For the other direction, if there is an assignment that satisfies at least $n + k$ clauses in $f(x)$, then for at least k values i it holds that both c_i^p and c_i^n are satisfied. For each such i, at least one of these two clauses is satisfied by a literal from c_i; hence,

the projection of this assignment to the variables of x satisfies at least k clauses in x. We conclude that an optimal solution to MAX-3-SAT satisfies k clauses if and only if an optimal solution to MAX-NM-SAT satisfies $n + k$ clauses.

Let y_{opt} be a set of clauses satisfied by an optimal solution to the MAX-3-SAT problem, and let y'_{opt} be an optimal solution to the MAX-NM-SAT problem. It is known that there is always an assignment that satisfies at least half of the clauses in the formula; hence, it holds that $\frac{1}{2}n \leq |y_{\text{opt}}|$. Now, for $\beta > 1$, let y' be a β-optimal solution to MAX-NM-SAT. Then, it holds that $|y'_{\text{opt}}| \leq \beta|y'|$. We will now show that if $\beta = \frac{3\alpha}{2\alpha+1}$, then $g(x, y')$ is an α-optimal solution for MAX-3-SAT. Let y be the set of clauses satisfied by the assignment $g(x, y')$:

$$|y| \geq |y'| - n \geq \frac{1}{\beta}|y'_{\text{opt}}| - n = \frac{1}{\beta}(|y_{\text{opt}}| + n) - n = \frac{1}{\beta}|y_{\text{opt}}| - \left(1 - \frac{1}{\beta}\right)n$$
$$\geq \frac{1}{\beta}|y_{\text{opt}}| - \left(1 - \frac{1}{\beta}\right)2|y_{\text{opt}}| = \left(\frac{3}{\beta} - 2\right)|y_{\text{opt}}|.$$

Hence, we have that

$$\frac{|y|}{|y_{\text{opt}}|} \geq \frac{3}{\beta} - 2 = \frac{1}{\alpha}$$
$$\Rightarrow \beta = \frac{3\alpha}{2\alpha + 1}.$$

Therefore, to obtain an α-optimal solution to MAX-3-SAT, we need to find a $\frac{3\alpha}{2\alpha+1}$-optimal solution to MAX-NM-SAT. □

Next, we construct a PTAS reduction from MAX-NM-SAT to our problem and this will conclude our proof of APX-completeness for $\Delta_{AB\to C\to B}$. Note that in our proof of hardness for MAX-NM-SAT, we constructed an instance in which every clause contains at most four literals; hence, we use this assumption in the proof of the following lemma.

Lemma 4.12. *Computing an optimal S-repair for $\Delta_{AB\to C\to B}$ is APX-complete.*

Proof. We construct a PTAS reduction (f, g, k) from MAX-NM-SAT to the problem of computing an optimal S-repair for $\Delta_{AB\to C\to B}$. In our construction, each variable z in a clause c will be represented by a tuple (c, b, z) (where $b = 1$ if c is positive, and $b = 0$ otherwise). The FD $C \to B$ will ensure that an assignment corresponding to a consistent subset is a valid truth assignment and the FD $AB \to C$ will ensure that each consistent subset contains at most one tuple for each clause. More formally, the input to the first problem is a formula ψ with the propositional variables $x_1, \ldots, x_n$, such that ψ has the form $c_1 \wedge \cdots \wedge c_m$ where each c_j is a clause. Each clause is a disjunction of literals from one of the following sets: *(a)* $\{x_i : i = 1, \ldots, n\}$ or *(b)* $\{\neg x_i : i = 1, \ldots, n\}$ (that is, each clause either contains only positive literals or only negative literals). The goal is to determine what is the maximum number of clauses in the formula ψ that can be simultaneously satisfied. Given such an input, we will construct the input $f(x) = T$ for our problem as follows. For each $i = 1, \ldots, n$ and $j = 1, \ldots, m$, T will contain the following tuples:

- $(c_j, 1, x_i)$, if c_j contains only positive variables and x_i appears in c_j.
- $(c_j, 0, x_i)$, if c_j contains only negative variables and $\neg x_i$ appears in c_j.

The weight of each tuple will be 1 (that is, T is an unweighted, duplicate-free table). We will now prove that there is an assignment that satisfies at least k clauses in ψ if and only if there is a consistent subset of the constructed table T that is obtained by deleting at most $N - k$ tuples, where N is the number of tuples in T.

The "if" direction. Assume that there is a consistent subset J of T that is obtained by deleting at most $N - k$ tuples (hence, J contains k tuples). The FD $AB \rightarrow C$ implies that no consistent subset of T contains two tuples (c_j, b_j, x_{i_1}) and (c_j, b_j, x_{i_2}) such that $x_{i_1} \neq x_{i_2}$. Thus, each consistent subset contains at most one tuple (c_j, b_j, x_i) for each c_j. We will now define an assignment τ as follows: $\tau(x_i) \stackrel{\text{def}}{=} b_j$ if there exists a tuple (c_j, b_j, x_i) in J for some c_j. Note that the FD $C \rightarrow B$ implies that no consistent subset contains two tuples $(c_{j_1}, 1, x_i)$ and $(c_{j_2}, 0, x_i)$; thus, the assignment is well defined. Finally, as mentioned above, J contains a tuple (c_j, b_j, x_i) for k clauses c_j from ψ. If x_i appears in c_j without negation, then it holds that $b_j = 1$; hence, $\tau(x_i) \stackrel{\text{def}}{=} 1$ and c_j is satisfied. Similarly, if x_i appears in c_j with negation, then it holds that $b_j = 0$; hence, $\tau(x_i) \stackrel{\text{def}}{=} 0$ and c_j is satisfied. Thus, each one of these k clauses is satisfied by τ, and we conclude that there exists an assignment that satisfies at least k clauses in ψ.

The "only if" direction. Assume that τ is an assignment that satisfies at least k clauses in ψ. We claim that there exists a consistent subset of T that is obtained by deleting $N - k$ tuples. Since τ satisfies at least k clauses, for each one of these clauses c_j there exists a variable $x_i \in c_j$, such that $\tau(x_i) = 1$ if x_i appears in c_j without negation or $\tau(x_i) = 0$ if it appears in c_j with negation. Let us build a consistent subset J as follows. For each c_j that is satisfied by τ, we will choose exactly one variable x_i that satisfies the above and add the tuple (c_j, b_j, x_i) (where $\tau(x_i) = b_j$) to J. Since there are at least k satisfied clauses, J will contain at least k tuples; thus, it is only left to prove that J is consistent. Let us assume, by way of contradiction, that J is not consistent. Since J contains one tuple for each satisfied c_j, no two tuples violate the FD $AB \rightarrow C$. Thus, J contains two tuples $(c_{j_1}, 1, x_i)$ and $(c_{j_2}, 0, x_i)$, but this is a contradiction to the fact that τ is an assignment (that is, it cannot be the case that $\tau(x_i) = 1$ and $\tau(x_i) = 0$). Therefore, J is a consistent subset of T that contains at least k tuples (hence, obtained by deleting at most $N - k$ tuples).

Hence, given a solution y' to our problem on $f(x)$, the solution $g(x, y')$ will assign the value 1 to the variable x_i if there is a tuple $(c_j, 1, x_i)$ in $T \setminus y'$, and it will assign the value 0 to the variable x_k if there is a tuple $(c_j, 0, x_k)$ in $T \setminus y'$. We will now prove that if y' is a $\frac{8\alpha-1}{7\alpha}$-optimal solution to our problem on $f(x)$, then $g(x, y')$ is an α-optimal solution to MAX-NM-SAT on x. Recall that we assumed that each clause in x contains at most four literals. Let n_i be the number of clauses in x that contain i literals for $i \in \{1, 2, 3, 4\}$. Then, the number of clauses in x is $n_1 + n_2 + n_3 + n_4$, and the number of tuples in the constructed table is $N = n_1 + 2n_2 + 3n_3 + 4n_4$. Let y_{opt} be a set of clauses satisfied by an optimal solution to MAX-NM-SAT on x and let y'_{opt} be an optimal solution to our problem. As we have already mentioned before, there is always an assignment that satisfies at least half of the clauses in the formula; hence, it holds that $|y_{\text{opt}}| \geq \frac{n_1+n_2+n_3+n_4}{2} = \frac{4n_1+4n_2+4n_3+4n_4}{8} \geq \frac{N}{8}$.

For $\beta > 1$, let y' be a β-optimal solution to our problem. Then,

$$|y'| \leq \beta |y'_{\text{opt}}|.$$

We will now show that if $\beta = \frac{8\alpha-1}{7\alpha}$, then $g(x, y')$ is an α-optimal solution for MAX-NM-SAT. Let y be the set of clauses satisfied by the assignment $g(x, y')$:

$$\begin{aligned} |y| &= N - |y'| \geq N - \beta |y'_{\text{opt}}| = N - \beta(N - |y_{\text{opt}}|) = \beta |y_{\text{opt}}| - (\beta - 1)N \\ &\geq \beta |y_{\text{opt}}| - (\beta - 1) \cdot 8 |y_{\text{opt}}| = (8 - 7\beta)|y_{\text{opt}}|. \end{aligned}$$

Hence, we have that

$$\begin{aligned} &\frac{|y|}{|y_{\text{opt}}|} \geq (8 - 7\beta) = \frac{1}{\alpha} \\ &\Rightarrow \beta = \frac{8\alpha - 1}{7\alpha}. \end{aligned}$$

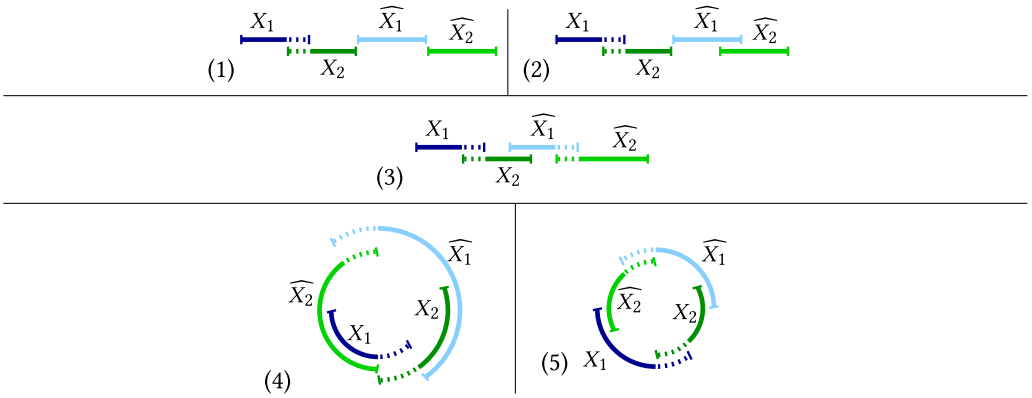

Fig. 5. Classes of FD sets that cannot be simplified.

Thus, to obtain an α-optimal solution to MAX-NM-SAT, we need to find a $\frac{8\alpha-1}{7\alpha}$-optimal solution to our problem. □

So far, we have shown that the problem of finding an optimal S-repair is APX-complete for all the FD sets in Table 1, and this concludes the first step of our hardness proof.

4.2.2 Step 2: Classifying FD Sets. We now show that when an FD set Δ cannot be further simplified, it can be classified to one of five classes of FD sets. Later, for each one of these classes, we will build a fact-wise reduction from one of the FD sets in Table 1 over the relation schema $R(A, B, C)$. We first identify that if an FD set Δ cannot be simplified, then there are at least two distinct local minima $X_1 \to Y_1$ and $X_2 \to Y_2$ in Δ. By a *local minimum*, we mean an FD with a set-minimal lhs, that is, an FD $X \to Y$ such that no FD $Z \to W$ in Δ satisfies that Z is a strict subset of X. We pick any two local minima from Δ. Then, we divide the FD sets into five classes based on the relationships between X_1, X_2, $cl_\Delta(X_1) \setminus X_1$, which we denote by $\widehat{X_1}$, and $cl_\Delta(X_2) \setminus X_2$, which we denote by $\widehat{X_2}$. The classes are illustrated in Figure 5.

Each line in Figure 5 represents one of X_1, X_2, $\widehat{X_1}$ and $\widehat{X_2}$. If two lines do not overlap, then the corresponding two sets are assumed to be disjoint. For example, the sets $\widehat{X_1}$ and $\widehat{X_2}$ in class (1) are disjoint. Overlapping lines represent sets that have a nonempty intersection, an example being $\widehat{X_1}$ and $\widehat{X_2}$ in class (2). When two dashed lines overlap, we do not assume anything about their intersection. As an example, the sets X_1 and X_2 can have an empty or a nonempty intersection in each of the classes. Finally, if a line covers another line, then the set that corresponds to the first line contains that of the second line. For instance, the set $\widehat{X_2}$ in class (4) contains the set $X_1 \setminus X_2$, while in class (5) it holds that $(X_1 \setminus X_2) \not\subseteq \widehat{X_2}$. We remark that Figure 5 covers the important cases that we need to analyze, but it misses a few cases. The full details and the proofs are given after the following example.

Example 4.13. In what follows, we discuss the five classes of Figure 5. Specifically, we mention which FD set of Table 1 is used in the corresponding fact-wise reductions, and give an example of an FD set in the class.

Class 1. We build fact-wise reductions from $\Delta_{A\to C\leftarrow B}$ to the FD sets in this class. An example of an FD set in this class is $\Delta_1 = \{A \to B, C \to D\}$. In this case $X_1 = \{A\}$, $X_2 = \{C\}$, $\widehat{X_1} = \{B\}$ and $\widehat{X_2} = \{D\}$. Thus, $\widehat{X_1} \cap X_2 = \emptyset$, $\widehat{X_2} \cap X_1 = \emptyset$ and $\widehat{X_1} \cap \widehat{X_2} = \emptyset$ and indeed the only overlapping lines in (1) are the dashed lines corresponding to X_1 and X_2.

Class 2. For this class, the fact-wise reductions are from $\Delta_{A\to B\to C}$. The FD set $\Delta_2 = \{A \to C, A \to D, B \to C, B \to E\}$, for example, belongs to this class. It holds that $X_1 = \{A\}$, $X_2 = \{B\}$, $\widehat{X_1} = \{C, D\}$ and $\widehat{X_2} = \{C, E\}$. Hence, $\widehat{X_1} \cap X_2 = \emptyset$ and $\widehat{X_2} \cap X_1 = \emptyset$, but $\widehat{X_1} \cap \widehat{X_2} \neq \emptyset$, and the difference from (1) is that the lines corresponding to $\widehat{X_1}$ and $\widehat{X_2}$ in (2) overlap.

Class 3. We again build fact-wise reductions from $\Delta_{A\to B\to C}$ to the FD sets in this class. As an example of an FD set in this class, we use the set $\Delta_3 = \{A \to B, A \to C, B \to D\}$. Here, it holds that $X_1 = \{A\}$, $X_2 = \{B\}$, $\widehat{X_1} = \{B, C, D\}$ and $\widehat{X_2} = \{D\}$. Thus, $\widehat{X_1} \cap X_2 \neq \emptyset$, but $\widehat{X_2} \cap X_1 = \emptyset$. The difference from (2) is that now the lines corresponding to X_2 and $\widehat{X_1}$ overlap, and we do not assume anything about the intersection between $\widehat{X_1}$ and $\widehat{X_2}$.

Class 4. Here, the fact-wise reductions are from $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$. An example of an FD set that belongs to this class is $\Delta_4 = \{A \to B, B \to C, C \to A\}$. In this case, we have three local minima. We pick two of them: $A \to B$ and $B \to C$. Now, $X_1 = \{A\}, X_2 = \{B\}, \widehat{X_1} = \{B, C\}$ and $\widehat{X_2} = \{A, C\}$. Thus, $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$. The difference from (3) is that now the lines corresponding to X_1 and $\widehat{X_2}$ overlap. Moreover, the line corresponding to $\widehat{X_1}$ covers the entire line corresponding to $X_2 \setminus X_1$ and the line corresponding to $\widehat{X_2}$ covers the entire line corresponding to $X_1 \setminus X_2$. This means that we assume that $(X_1 \setminus X_2) \subseteq \widehat{X_2}$ and $(X_2 \setminus X_1) \subseteq \widehat{X_1}$.

Class 5. For FD sets in this class, we build a fact-wise reduction from $\Delta_{AB\to C\to B}$. The FD set $\Delta_5 = \{AB \to C, C \to A, C \to D\}$ is an example of an FD set that belongs to this class. Here, $X_1 = \{A, B\}$, $X_2 = \{C\}, \widehat{X_1} = \{C, D\}$ and $\widehat{X_2} = \{A, D\}$, therefore $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$. The difference from (4) is that now we assume that $(X_1 \setminus X_2) \not\subseteq \widehat{X_2}$.

Note that for each class, we build infinitely many fact-wise reductions, one for each FD set in this class.

We now show that an FD set that cannot be simplified indeed belongs to one of the five classes. First, we prove the following.

Lemma 4.14. *Let Δ be an FD set that cannot be simplified. Then, Δ contains at least two local minima $X_1 \to Y_1$ and $X_2 \to Y_2$, such that $X_1 \neq X_2$.*

Proof. Assume, by way of contradiction, that Δ does not contain two distinct local minima. In this case, Δ contain exactly one local minimum $X \to A$. That is, for every FD $Z \to B$ there exists an FD $Z' \to B'$ such that $Z' \subset Z$, except for FDs of the form $X \to A'$. Hence, for every FD $Z \to B$ in Δ it holds that $X \subseteq Z$. In this case, Δ either has a consensus FD (when $X = \emptyset$) or a common lhs (when $X \neq \emptyset$), which is a contradiction to the fact that Δ cannot be simplified. □

Let Δ be an FD set that cannot be simplified, and let $X_1 \to Y_1$ and $X_2 \to Y_2$ be two local minima in Δ. One of the following holds:

(1) $\widehat{X_1} \cap cl_\Delta(X_2) = \emptyset$ and $\widehat{X_2} \cap cl_\Delta(X_1) = \emptyset$.
(2) $\widehat{X_1} \cap \widehat{X_2} \neq \emptyset$, $\widehat{X_1} \cap X_2 = \emptyset$ and $\widehat{X_2} \cap X_1 = \emptyset$.
(3) $\widehat{X_1} \cap X_2 \neq \emptyset$ or $\widehat{X_2} \cap X_1 \neq \emptyset$, but not both.
(4) $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$ and also $(X_1 \setminus X_2) \subseteq \widehat{X_2}$ and $(X_2 \setminus X_1) \subseteq \widehat{X_1}$.
(5) $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$ and also either $(X_2 \setminus X_1) \not\subseteq \widehat{X_1}$ or $(X_1 \setminus X_2) \not\subseteq \widehat{X_2}$ (or both).

In all of the above cases, we assume nothing about the intersection between X_1 and X_2 (that is, the intersection may be empty or nonempty), but we do assume something about the relationship between the rest of the attributes in the closure of X_1 under Δ (that is, the attributes in $\widehat{X_1}$) and $cl_\Delta(X_2)$, and also about the relationship between the rest of the attributes in the closure of X_2 under Δ (that is, the attributes in $\widehat{X_2}$) and $cl_\Delta(X_1)$.

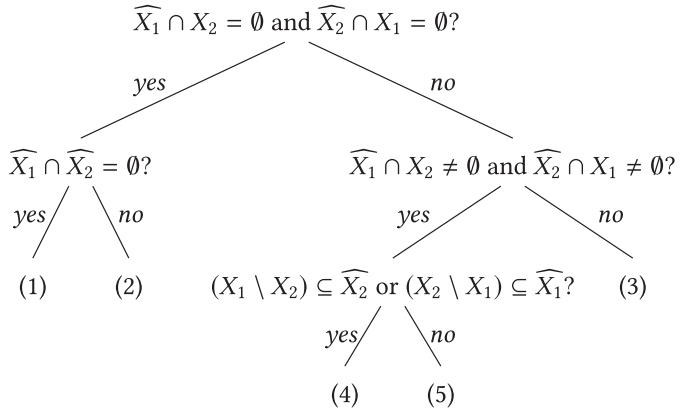

Fig. 6. Classification according to the five conditions on Δ, X_1, and X_2.

For the first two cases, we assume that there is no intersection between $\widehat{X_1}$ and X_2 and similarly there is no intersection between $\widehat{X_2}$ and X_1. The difference between the cases is that in the first case, we also assume that there is no intersection between $\widehat{X_1}$ and $\widehat{X_2}$, while in the second case, we assume that this intersection is nonempty. For the other three cases, we assume that at least one of the intersections $\widehat{X_1} \cap X_2$ or $\widehat{X_2} \cap X_1$ is nonempty.

In the third case, we assume that only one of the above intersections is nonempty; that is, either $\widehat{X_1} \cap X_2 \neq \emptyset$ or $\widehat{X_2} \cap X_1 \neq \emptyset$, but not both. For the last two cases, we assume that both intersections are not empty, and the difference between these cases is based on containment. In the fourth case, we assume that $(X_1 \setminus X_2) \subseteq \widehat{X_2}$ and $(X_2 \setminus X_1) \subseteq \widehat{X_1}$), while in the last case, we assume that at least one of these containments does not hold.

Using the decision tree of Figure 6, the reader can verify that the above five cases cover all the possible cases. Lemma 4.14 states that a set of FDs Δ that cannot be simplified contains at least two local minima $X_1 \to Y_1$ and $X_2 \to Y_2$; hence, Δ can be classified into one of the five classes mentioned above based on these two local minima.

4.2.3 Step 3: Hard Classes. Next, we prove that for each one of the five classes mentioned above, there is a fact-wise reduction from one of the hard schemas we discussed earlier (the schemas of Table 1). Lemma 4.4 will then imply that computing an optimal S-repair for a nontrivial set of FDs that cannot be simplified is APX-complete.

Note that in all of the fact-wise reductions, we assume that the set of FDs does not contain trivial FDs, since the OSRSucceeds algorithm removes trivial FDs from Δ at each iteration. We start with the first class.

Lemma 4.15. *Let R be a relation schema and let Δ be an FD set over R that does not contain trivial FDs. Suppose that Δ contains two distinct local minima $X_1 \to Y_1$ and $X_2 \to Y_2$, and the following hold:*

- $\widehat{X_1} \cap cl_\Delta(X_2) = \emptyset$,
- $\widehat{X_2} \cap cl_\Delta(X_1) = \emptyset$.

Then, there is a fact-wise reduction from $(R(A, B, C), \Delta_{A \to C \leftarrow B})$ to (R, Δ).

Proof. We define a fact-wise reduction $\Pi : (R(A, B, C), \Delta_{A \to C \leftarrow B}) \to (R, \Delta)$, using the FDs $X_1 \to Y_1$ and $X_2 \to Y_2$ and the constant $\odot \in$ Const. Let $\mathbf{t} = (a, b, c)$ be a tuple over $R(A, B, C)$ and let

$\{A_1, \ldots, A_n\}$ be the set of attributes in R. We define Π as follows:

$$\Pi(\mathbf{t}).A_k \stackrel{\text{def}}{=} \begin{cases} \odot & A_k \in X_1 \cap X_2 \\ a & A_k \in X_1 \setminus X_2 \\ b & A_k \in X_2 \setminus X_1 \\ \langle a, c\rangle & A_k \in \widehat{X_1} \\ \langle b, c\rangle & A_k \in \widehat{X_2} \\ \langle a, b\rangle & \text{otherwise.} \end{cases}$$

It is left to show that Π is a fact-wise reduction. To do so, we prove that Π is well defined, injective and preserves consistency and inconsistency.

Π is well defined. This is straightforward from the definition and the fact that $\widehat{X_1} \cap cl_\Delta(X_2) = \emptyset$ and $\widehat{X_2} \cap cl_\Delta(X_1) = \emptyset$.

Π is injective. Let $\mathbf{t}, \mathbf{t}'$ be two distinct tuples, such that $\mathbf{t} = (a, b, c)$ and $\mathbf{t}' = (a', b', c')$ (and $\mathbf{t} \neq \mathbf{t}'$). Assume that $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$. Let us denote $\Pi(\mathbf{t}) = (x_1, \ldots, x_n)$ and $\Pi(\mathbf{t}') = (x'_1, \ldots, x'_n)$. Note that $X_1 \setminus X_2$ and $X_2 \setminus X_1$ are not empty, since $X_1 \neq X_2$. Moreover, since both FDs are minimal, $X_1 \not\subset X_2$ and $X_2 \not\subset X_1$. Therefore, there are l and p such that $\Pi(\mathbf{t}).A_l = a$, $\Pi(\mathbf{t}).A_p = b$. Furthermore, since $X_1 \to Y_1$ and $X_2 \to Y_2$ are not trivial, there are m and n such that $\Pi(\mathbf{t}).A_m = \langle a, c\rangle$ and $\Pi(\mathbf{t}).A_n = \langle b, c\rangle$. Similarly, it holds that $\Pi(\mathbf{t}').A_l = a'$, $\Pi(\mathbf{t}').A_p = b'$, $\Pi(\mathbf{t}').A_m = \langle a', c'\rangle$, and $\Pi(\mathbf{t}').A_n = \langle b', c'\rangle$. Hence, $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$ implies that $\Pi(\mathbf{t}).A_l = \Pi(\mathbf{t}').A_l$, $\Pi(\mathbf{t}).A_p = \Pi(\mathbf{t}').A_p$, $\Pi(\mathbf{t}).A_m = \Pi(\mathbf{t}').A_m$ and also $\Pi(\mathbf{t}).A_n = \Pi(\mathbf{t}').A_n$. We obtain that $a = a'$, $b = b'$ and $c = c'$, which implies $\mathbf{t} = \mathbf{t}'$.

Π preserves consistency. Let $\mathbf{t} = (a, b, c)$ and $\mathbf{t}' = (a', b', c')$ be two distinct tuples. We contend that the pair $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t. $\Delta_{A\to C\leftarrow B}$ if and only if the set $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

The "if" direction. Assume $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{A\to C\leftarrow B}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ. Since $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{A\to C\leftarrow B}$ it either holds that $a = a'$ and $c \neq c'$ or $b = b'$ and $c \neq c'$ (or both). In the first case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on the attributes on the left-hand side of the FD $X_1 \to Y_1$, but do not agree on the attribute in Y_1 (since the FD is not trivial). Similarly, in the second case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on the attributes on the left-hand side of the FD $X_2 \to Y_2$, but do not agree on the attribute in Y_2. Thus, $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ does not satisfy at least one of these FDs and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ. This concludes our proof of the "if" direction.

The "only if" direction. Assume that $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t $\Delta_{A\to C\leftarrow B}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t Δ. First, note that each FD that contains an attribute $A_k \notin (cl_\Delta(X_1) \cup cl_\Delta(X_2))$ on its left-hand side is satisfied by $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$, since $\mathbf{t}$ and $\mathbf{t}'$ cannot agree on both A and B (otherwise, the FD $A \to C$ implies that $\mathbf{t} = \mathbf{t}'$). Thus, from now on, we will only consider FDs that do not contain an attribute $A_k \notin (cl_\Delta(X_1) \cup cl_\Delta(X_2))$ on their left-hand side. The FDs in $\Delta_{A\to C\leftarrow B}$ imply that if $\mathbf{t}$ and $\mathbf{t}'$ agree on one of $\{A, B\}$ then they also agree on C; thus, one of the following holds:

(1) $a \neq a'$, $b = b'$ and $c = c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ only agree on the attributes A_k such that $A_k \in X_1 \cap X_2$ or $A_k \in X_2 \setminus X_1$ or $A_k \in cl_\Delta(X_2) \setminus X_2$. That is, they only agree on the attributes A_k such that $A_k \in cl_\Delta(X_2)$. Thus, each FD that contains an attribute $A_k \notin cl_\Delta(X_2)$ on its left-hand side is satisfied. Moreover, any FD that contains only attributes $A_k \in cl_\Delta(X_2)$ on its left-hand side, also contains only attributes $A_k \in cl_\Delta(X_2)$ on its right-hand side (by definition of a closure). Thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on both the left-hand side and the right-hand side of such FDs and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ satisfies all the FDs in Δ.

(2) $a = a'$, $b \neq b'$ and $c = c'$. This case is symmetric to the previous one; thus, a similar proof applies for this case as well.
(3) $a \neq a'$, $b \neq b'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ only agree on the attributes A_k such that $A_k \in X_1 \cap X_2$. Since $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$ are local minima, there is no FD in Δ that contains only attributes A_k such that $A_k \in X_1 \cap X_2$ on its left-hand side (as if there is an FD $Z \rightarrow B$ in Δ, such that $Z \subseteq X_1 \cap X_2$, then $Z \subset X_1$ in contradiction to the fact that X_1 is a local minimum). Hence, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ do not agree on the left-hand side of any FD in Δ and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

This concludes our proof of the "only if" direction. □

Next, we consider the second and the third classes, as one fact-wise reduction works for both.

Lemma 4.16. *Let R be a relation schema and let Δ be an FD set over R that does not contain trivial FDs. Suppose that Δ contains two distinct local minima $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$, and one of the following holds:*

- $\widehat{X_1} \cap \widehat{X_2} \neq \emptyset$, $\widehat{X_1} \cap X_2 = \emptyset$ *and* $\widehat{X_2} \cap X_1 = \emptyset$,
- $\widehat{X_1} \cap X_2 \neq \emptyset$ *and* $\widehat{X_2} \cap X_1 = \emptyset$.

Then, there is a fact-wise reduction from $(R(A, B, C), \Delta_{A \rightarrow B \rightarrow C})$ to (R, Δ).

Proof. We define a fact-wise reduction $\Pi : (R(A, B, C), \Delta_{A \rightarrow B \rightarrow C}) \rightarrow (R, \Delta)$, using $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$ and the constant $\odot \in \text{Const}$. Let $\mathbf{t} = (a, b, c)$ be a tuple over $R(A, B, C)$ and let $\{A_1, \ldots, A_n\}$ be the set of attributes in R. We define Π as follows:

$$\Pi(\mathbf{t}).A_k \stackrel{\text{def}}{=} \begin{cases} \odot & A_k \in X_1 \cap X_2 \\ a & A_k \in X_1 \setminus X_2 \\ b & A_k \in X_2 \setminus X_1 \\ \langle a, c \rangle & A_k \in \widehat{X_1} \setminus cl_\Delta(X_2) \\ \langle b, c \rangle & A_k \in \widehat{X_2} \\ a & \text{otherwise.} \end{cases}$$

It is left to show that Π is a fact-wise reduction. To do so, we prove that Π is well defined, injective, and preserves consistency and inconsistency.

Π is well defined. This is straightforward from the definition and the fact that $\widehat{X_2} \cap X_1 = \emptyset$ in both cases.

Π is injective. Let $\mathbf{t}, \mathbf{t}'$ be two distinct tuples, such that $\mathbf{t} = (a, b, c)$ and $\mathbf{t}' = (a', b', c')$ (and $\mathbf{t} \neq \mathbf{t}'$). Assume that $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$. Let us denote $\Pi(\mathbf{t}) = (x_1, \ldots, x_n)$ and $\Pi(\mathbf{t}') = (x'_1, \ldots, x'_n)$. Note that $X_1 \setminus X_2$ and $X_2 \setminus X_1$ are not empty, since $X_1 \neq X_2$. Moreover, since both FDs are minimal, $X_1 \not\subset X_2$ and $X_2 \not\subset X_1$. Therefore, there are l and p such that $\Pi(\mathbf{t}).A_l = a$, $\Pi(\mathbf{t}).A_p = b$. Furthermore, since $X_2 \rightarrow Y_2$ is not trivial, there is at least one m such that $\Pi(\mathbf{t}).A_m = \langle b, c \rangle$. Similarly, it holds that $\Pi(\mathbf{t}').A_l = a'$, $\Pi(\mathbf{t}').A_p = b'$, and $\Pi(\mathbf{t}').A_m = \langle b', c' \rangle$. Hence, $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$ implies that $\Pi(\mathbf{t}).A_l = \Pi(\mathbf{t}').A_l$, $\Pi(\mathbf{t}).A_p = \Pi(\mathbf{t}').A_p$ and $\Pi(\mathbf{t}).A_m = \Pi(\mathbf{t}').A_m$. We obtain that $a = a'$, $b = b'$ and $c = c'$, which implies $\mathbf{t} = \mathbf{t}'$.

Π preserves consistency. Let $\mathbf{t} = (a, b, c)$ and $\mathbf{t}' = (a', b', c')$ be two distinct tuples. We contend that the pair $\{t, t'\}$ is consistent w.r.t. $\Delta_{A \rightarrow B \rightarrow C}$ if and only if the set $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

The "if" direction. Assume $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{A \rightarrow B \rightarrow C}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ. Since $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{A \rightarrow B \rightarrow C}$, one of the following holds:

(1) $a = a'$ and $b \neq b'$. For the first case of this lemma, since $\widehat{X_1} \cap \widehat{X_2} \neq \emptyset$, at least one attribute $A_k \in \widehat{X_1}$ also belongs to $\widehat{X_2}$ and it holds that $\Pi(\mathbf{t}).A_k = \langle b, c\rangle$ and $\Pi(\mathbf{t}').A_k = \langle b', c'\rangle$. For the second case of this lemma, since $\widehat{X_1} \cap X_2 \neq \emptyset$, at least one attribute $A_k \in \widehat{X_1}$ also belongs to X_2 and it holds that $\Pi(\mathbf{t}).A_k = b$ and $\Pi(\mathbf{t}').A_k = b'$. Moreover, in both cases, by definition of a closure, the FD $X_1 \rightarrow A_k$ is implied by Δ. Hence, in both cases, the tuples $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on the attributes on the left-hand side of the FD $X_1 \rightarrow A_k$, but do not agree on the right-hand side of this FD. If two tuples do not satisfy an FD that is implied by a set Δ of FDs, then they also do not satisfy Δ; thus, $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ.

(2) $a = a'$, $b = b'$ and $c \neq c'$. For the first case of this lemma, as mentioned above, there is an attribute $A_k \in \widehat{X_1}$ such that $\Pi(\mathbf{t}).A_k = \langle b, c\rangle$ and $\Pi(\mathbf{t}').A_k = \langle b', c'\rangle$. Moreover, by definition of a closure, the FD $X_1 \rightarrow A_k$ is implied by Δ. The tuples $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on the attributes on the left-hand side of the FD $X_1 \rightarrow A_k$, but do not agree on the right-hand side of this FD. For the second case of this lemma, since it holds that $\widehat{X_2} \cap X_1 = \emptyset$, and since the FD $X_2 \rightarrow Y_2$ is not trivial, there is at least one attribute $A_k \in \widehat{X_2}$ such that $\Pi(\mathbf{t}).A_k = \langle b, c\rangle$ and $\Pi(\mathbf{t}').A_k = \langle b', c'\rangle$. Furthermore, the FDs $X_2 \rightarrow A_k$ is implied by Δ. The tuples $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ again agree on the attributes on the left-hand side of the FD $X_2 \rightarrow A_k$, but do not agree on the right-hand side of this FD. In both cases, there exist two tuples that do not satisfy an FD that is implied by Δ; thus, they also do not satisfy Δ, and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ.

(3) $a \neq a'$, $b = b'$ and $c \neq c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on the attributes on the left-hand side of the FD $X_2 \rightarrow Y_2$, but do not agree on the right-hand side of this FD (since the FD is not trivial and contains at least one attribute A_k such that $\Pi(\mathbf{t}).A_k = \langle b, c\rangle$ and $\Pi(\mathbf{t}').A_k = \langle b', c'\rangle$ on its right-hand side). Therefore, $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ.

This concludes our proof of the "if" direction.

The "only if" direction. Assume that $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t $\Delta_{A\rightarrow B\rightarrow C}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t Δ. First, note that each FD that contains an attribute $A_k \notin (cl_\Delta(X_1) \cup cl_\Delta(X_2))$ on its left-hand side is satisfied by $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$, since $\mathbf{t}$ and $\mathbf{t}'$ cannot agree on A (otherwise, the FDs $A \rightarrow B$ and $B \rightarrow C$ imply that $\mathbf{t} = \mathbf{t}'$). Thus, from now on, we will only consider FDs that do not contain an attribute $A_k \notin (cl_\Delta(X_1) \cup cl_\Delta(X_2))$ on their left-hand side. One of the following holds:

(1) $a \neq a'$, $b = b'$ and $c = c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ only agree on the attributes A_k such that $A_k \in X_1 \cap X_2$ or $A_k \in X_2 \setminus X_1$ or $A_k \in \widehat{X_2}$. That is, they only agree on the attributes A_k such that $A_k \in cl_\Delta(X_2)$. Thus, each FD that contains an attribute $A_k \notin cl_\Delta(X_2)$ on its left-hand side is satisfied. Moreover, any FD that contains only attributes $A_k \in cl_\Delta(X_2)$ on its left-hand side, also contains only attributes $A_k \in cl_\Delta(X_2)$ on its right-hand side (by definition of a closure); thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on both the left-hand side and the right-hand side of such FDs and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ satisfies all the FDs in Δ.

(2) $a \neq a'$, $b \neq b'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ only agree on the attributes A_k such that $A_k \in X_1 \cap X_2$. Since $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$ are minimal, there is no FD in Δ that contains only attributes A_k such that $A_k \in X_1 \cap X_2$ on its left-hand side. Thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ do not agree on the left-hand side of any FD in Δ and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

This concludes our proof of the "only if" direction. □

We now consider the fourth class. For this class, we assume that $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$ and also that $(X_1 \setminus X_2) \subseteq \widehat{X_2}$ and $(X_2 \setminus X_1) \subseteq \widehat{X_1}$. In this case, Δ contains at least one more local minimum (that is, there are at least three). Otherwise, for every FD $Z \rightarrow B$ in Δ it holds that either $X_1 \subseteq Z$ or $X_2 \subseteq Z$. If $X_1 \cap X_2 \neq \emptyset$, then Δ contains a common lhs (an attribute from $X_1 \cap X_2$). If

$X_1 \cap X_2 = \emptyset$, then Δ contains an lhs marriage (because $X_1 \subseteq \widehat{X_2}$ and $X_2 \subseteq \widehat{X_1}$, which means that $cl_\Delta(X_1) = cl_\Delta(X_2)$).

Lemma 4.17. *Let R be a relation schema and let Δ be an FD set over R that does not contain trivial FDs. Suppose that Δ contains three distinct local minima $X_1 \to Y_1, X_2 \to Y_2$ and $X_3 \to Y_3$. Then, there is a fact-wise reduction from $(R(A,B,C), \Delta_{AB\leftrightarrow AC\leftrightarrow BC})$ to (R,Δ).*

Proof. We define a fact-wise reduction $\Pi : (R(A,B,C), \Delta_{AB\leftrightarrow AC\leftrightarrow BC}) \to (R,\Delta)$, using $X_1 \to Y_1$, $X_2 \to Y_2$ and $X_3 \to Y_3$ and the constant $\odot \in \mathsf{Const}$. Let $\mathbf{t} = (a,b,c)$ be a tuple over $R(A,B,C)$ and let $\{A_1,\ldots,A_n\}$ be the set of attributes in R. We define Π as follows:

$$\Pi(\mathbf{t}).A_k \stackrel{\text{def}}{=} \begin{cases} \odot & A_k \in X_1 \cap X_2 \cap X_3 \\ a & A_k \in (X_1 \cap X_2) \setminus X_3 \\ b & A_k \in (X_1 \cap X_3) \setminus X_2 \\ c & A_k \in (X_2 \cap X_3) \setminus X_1 \\ \langle a,b\rangle & A_k \in X_1 \setminus X_2 \setminus X_3 \\ \langle a,c\rangle & A_k \in X_2 \setminus X_1 \setminus X_3 \\ \langle b,c\rangle & A_k \in X_3 \setminus X_1 \setminus X_2 \\ \langle a,b,c\rangle & \text{otherwise.} \end{cases}$$

It is left to show that Π is a fact-wise reduction. To do so, we prove that Π is well defined, injective, and preserves consistency and inconsistency.

Π is well defined. This is straightforward from the definition.

Π is injective. Let $\mathbf{t},\mathbf{t}'$ be two distinct tuples, such that $\mathbf{t} = (a,b,c)$ and $\mathbf{t}' = (a',b',c')$ (and $\mathbf{t} \neq \mathbf{t}'$). Assume that $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$. Let us denote $\Pi(\mathbf{t}) = (x_1,\ldots,x_n)$ and $\Pi(\mathbf{t}') = (x'_1,\ldots,x'_n)$. Note that X_1 contains at least one attribute that does not belong to X_3 (otherwise, it holds that $X_1 \subseteq X_3$, which is a contradiction to the fact that X_3 is minimal). Thus, there exists an attribute A_l such that either $\Pi(\mathbf{t}).A_l = a$ and $\Pi(\mathbf{t}').A_l = a'$ or $\Pi(\mathbf{t}).A_l = \langle a,b\rangle$ and $\Pi(\mathbf{t}').A_l = \langle a',b'\rangle$. Similarly, X_3 contains at least one attribute that does not belong to X_2. Thus, there exists an attribute A_p such that either $\Pi(\mathbf{t}).A_p = b$ and $\Pi(\mathbf{t}').A_p = b'$ or $\Pi(\mathbf{t}).A_p = \langle b,c\rangle$ and $\Pi(\mathbf{t}').A_p = \langle b',c'\rangle$. Finally, X_2 contains at least one attribute that does not belong to X_1. Thus, there exists an attribute A_r such that either $\Pi(\mathbf{t}).A_r = c$ and $\Pi(\mathbf{t}').A_r = c'$ or $\Pi(\mathbf{t}).A_r = \langle a,c\rangle$ and $\Pi(\mathbf{t}').A_r = \langle a',c'\rangle$. Hence, $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$ implies that $\Pi(\mathbf{t}).A_l = \Pi(\mathbf{t}').A_l$, $\Pi(\mathbf{t}).A_p = \Pi(\mathbf{t}').A_p$ and $\Pi(\mathbf{t}).A_r = \Pi(\mathbf{t}').A_r$. We obtain that $a = a'$, $b = b'$ and $c = c'$, which implies $\mathbf{t} = \mathbf{t}'$.

Π preserves consistency. Let $\mathbf{t} = (a,b,c)$ and $\mathbf{t}' = (a',b',c')$ be two distinct tuples. We contend that the set $\{t,t'\}$ is consistent w.r.t. $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$ if and only if the set $\{\Pi(\mathbf{t}),\Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

The "if" direction. Assume that $\{\mathbf{t},\mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$. We prove that $\{\Pi(\mathbf{t}),\Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ. Since $\{\mathbf{t},\mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$, $\mathbf{t}$ and $\mathbf{t}'$ agree on two attributes, but do not agree on the third one. Thus, one of the following holds:

- $a = a'$, $b = b'$, and $c \neq c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all of the attributes that appear on the left-hand side of $X_1 \to Y_1$. Since this FD is not trivial, it must contain on its right-hand side an attribute A_k such that $A_k \notin X_1$. That is, one of the following holds: *(a)* $\Pi(\mathbf{t}).A_k = c$,*(b)* $\Pi(\mathbf{t}).A_k = \langle a,c\rangle$, *(c)* $\Pi(\mathbf{t}).A_k = \langle b,c\rangle$, or *(d)* $\Pi(\mathbf{t}).A_k = \langle a,b,c\rangle$. Since $c \neq c'$, it holds that $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ do not satisfy the FD $X_1 \to Y_1$ and $\{\Pi(\mathbf{t}),\Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ.

- $a = a'$, $b \neq b'$, and $c = c'$. This case is symmetric to the first one. $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all of the attributes that appear on the left-hand side of $X_2 \rightarrow Y_2$, but do not agree on the attribute that appears on the right-hand side of the FD.
- $a \neq a'$, $b = b'$ and $c = c'$. This case is also symmetric to the first one. Here, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on the left-hand side, but not on the right-hand side of the FD $X_3 \rightarrow Y_3$.

The "only if" direction. Assume that $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t $\Delta_{AB \leftrightarrow AC \leftrightarrow BC}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t Δ. Note that $\mathbf{t}$ and $\mathbf{t}'$ cannot agree on more than one attribute (otherwise, they will violate at least one FD in $\Delta_{AB \leftrightarrow AC \leftrightarrow BC}$). Thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ may only agree on attributes that appear in $X_1 \cap X_2 \cap X_3$ and in one of $(X_1 \cap X_2) \setminus X_3$, $(X_1 \cap X_3) \setminus X_2$ or $(X_2 \cap X_3) \setminus X_1$. As mentioned above, X_1 contains at least one attribute that does not belong to X_3; thus, no FD in Δ contains only attributes from $X_1 \cap X_2 \cap X_3$ and $(X_1 \cap X_3) \setminus X_2$ on its left-hand side (otherwise, X_1 will not be minimal). Similarly, no FD in Δ contains only attributes from $X_1 \cap X_2 \cap X_3$ and $(X_2 \cap X_3) \setminus X_1$ on its left-hand side, and no FD in Δ contains only attributes from $X_1 \cap X_2 \cap X_3$ and $(X_1 \cap X_2) \setminus X_3$ on its left-hand side. Therefore $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ do not agree on the left-hand side of any FD in Δ, and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ. □

Finally, we consider the fifth and last class.

Lemma 4.18. *Let R be a relation schema and let Δ be an FD set over R that does not contain trivial FDs. Suppose that Δ contains two distinct local minima $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$, and the following hold:*

- $\widehat{X_1} \cap X_2 \neq \emptyset$ *and* $\widehat{X_2} \cap X_1 \neq \emptyset$,
- $(X_2 \setminus X_1) \not\subseteq \widehat{X_1}$.

Then, there is a fact-wise reduction from $(R(A, B, C), \Delta_{AB \rightarrow C \rightarrow B})$ to (R, Δ).

Proof. We define a fact-wise reduction $\Pi : (R(A, B, C), \Delta_{AB \rightarrow C \rightarrow B}) \rightarrow (R, \Delta)$, using $X_1 \rightarrow Y_1$, $X_2 \rightarrow Y_2$ and the constant $\odot \in \mathrm{Const}$. Let $\mathbf{t} = (a, b, c)$ be a tuple over $R(A, B, C)$ and let $\{A_1, \ldots, A_n\}$ be the set of attributes in R. We define Π as follows:

$$\Pi(\mathbf{t}).A_k \stackrel{\text{def}}{=} \begin{cases} \odot & A_k \in X_1 \cap X_2 \\ c & A_k \in X_1 \setminus X_2 \\ b & A_k \in (X_2 \setminus X_1) \cap \widehat{X_1} \\ \langle a, b \rangle & A_k \in (X_2 \setminus X_1) \setminus \widehat{X_1} \\ \langle b, c \rangle & A_k \in \widehat{X_1} \setminus (X_2 \setminus X_1) \\ \langle a, b, c \rangle & \text{otherwise.} \end{cases}$$

It is left to show that Π is a fact-wise reduction. To do so, we prove that Π is well defined, injective, and preserves consistency and inconsistency.

Π is well defined. This is straightforward from the definition.

Π is injective. Let $\mathbf{t}, \mathbf{t}'$ be two distinct tuples, such that $\mathbf{t} = (a, b, c)$ and $\mathbf{t}' = (a', b', c')$ (and $\mathbf{t} \neq \mathbf{t}'$). Assume that $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$. Let us denote $\Pi(\mathbf{t}) = (x_1, \ldots, x_n)$ and $\Pi(\mathbf{t}') = (x'_1, \ldots, x'_n)$. Since the FD $X_2 \rightarrow Y_2$ is a local minimum, it holds that $X_1 \not\subseteq X_2$. Thus, there is an attribute that appears in X_1, but does not appear in X_2. Moreover, it holds that $(X_2 \setminus X_1) \not\subseteq (cl_\Delta(X_1) \setminus X_1)$; thus, $X_2 \setminus X_1$ contains at least one attribute that does not appear in $cl_\Delta(X_1) \setminus X_1$. Therefore, there are l and p such that $\Pi(\mathbf{t}).A_l = c$ and $\Pi(\mathbf{t}').A_l = c'$, and also $\Pi(\mathbf{t}).A_p = \langle a, b \rangle$ and $\Pi(\mathbf{t}').A_p = \langle a', b' \rangle$. Hence, $\Pi(\mathbf{t}) = \Pi(\mathbf{t}')$ implies that $\Pi(\mathbf{t}).A_l = \Pi(\mathbf{t}').A_l$ and $\Pi(\mathbf{t}).A_p = \Pi(\mathbf{t}').A_p$. We obtain that $a = a'$, $b = b'$ and $c = c'$, which implies $\mathbf{t} = \mathbf{t}'$.

Π preserves consistency. Let $\mathbf{t} = (a, b, c)$ and $\mathbf{t}' = (a', b', c')$ be two distinct tuples. We contend that the set $\{t, t'\}$ is consistent w.r.t. $\Delta_{AB\to C\to B}$ if and only if the set $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

The "if" direction. Assume $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{AB\to C\to B}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ. Since $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta_{AB\to C\to B}$, one of the following holds:

(1) $a = a$, $b = b'$, and $c \neq c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all of the attributes that appear in X_2. Since $X_2 \to Y_2$ is not trivial, the attribute A_k in Y_2 does not belong to X_2. That is, one of the following holds: *(a)* $\Pi(\mathbf{t}).A_k = c$, *(b)* $\Pi(\mathbf{t}).A_k = \langle b, c\rangle$, or *(c)* $\Pi(\mathbf{t}).A_k = \langle a, b, c\rangle$. Since $c \neq c'$, it holds that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ violates the FD $X_2 \to Y_2$ and it is inconsistent w.r.t. Δ.
(2) $b \neq b'$ and $c = c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all of the attributes that appear in X_1. Since $X_1 \to Y_1$ is not trivial, the attribute A_k in Y_1 does not belong to X_1. That is, one of the following holds: *(a)* $\Pi(\mathbf{t}).A_k = b$, *(b)* $\Pi(\mathbf{t}).A_k = \langle a, b\rangle$, *(c)* $\Pi(\mathbf{t}).A_k = \langle b, c\rangle$, or *(d)* $\Pi(\mathbf{t}).A_k = \langle a, b, c\rangle$. Since $b \neq b'$, it holds that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ violates the FD $X_1 \to Y_1$ and it is again inconsistent w.r.t. Δ.

The "only if" direction. Assume that $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t $\Delta_{AB\to C\to B}$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t Δ. Note that $\mathbf{t}$ and $\mathbf{t}'$ cannot agree on the value of both attributes A and B, since if this is the case, the FD $AB \to C$ implies that they also agree on the value of attribute C, and $\mathbf{t} = \mathbf{t}'$. One of the following holds:

(1) $b \neq b'$ and $c \neq c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ only agree on the attributes A_k such that $A_k \in X_1 \cap X_2$. Since $X_1 \to Y_1$ and $X_2 \to Y_2$ are local minima, there is no FD in Δ that contains on its left-hand side only attributes A_k such that $A_k \in X_1 \cap X_2$. Thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ do not agree on the left-hand side of any FD in Δ and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.
(2) $a \neq a'$, $b = b'$, and $c = c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all of the attributes that belong to $cl_\Delta(X_1)$, and only on these attributes. Any FD in Δ that contains only attributes from $cl_\Delta(X_1)$ on its left-hand side, also contains only attributes from $cl_\Delta(X_1)$ on its right-hand side (by definition of closure). Thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ satisfy all the FDs in Δ.
(3) $a \neq a'$, $b = b'$, and $c \neq c'$. In this case, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ only agree on the attributes A_k such that $A_k \in X_1 \cap X_2$ or $A_k \in (X_2 \setminus X_1) \cap (cl_\Delta(X_1) \setminus X_1)$. Since the FD $X_2 \to Y_2$ is a local minimum, and since X_2 contains an attribute that does not belong to $cl_\Delta(X_1)$, no FD in Δ contains on its left-hand side only attributes A_k such that $A_k \in X_1 \cap X_2$ or $A_k \in (X_2 \setminus X_1) \cap (cl_\Delta(X_1) \setminus X_1)$. Thus, $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ do not agree on the left-hand side of any FD in Δ and $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ. □

We now prove that computing an optimal S-repair for a set of FDs that cannot be simplified is APX-complete.

Lemma 4.19. *Let R be a relation schema and let Δ be a nontrivial FD set over R. If no simplification can be applied to Δ, then computing an optimal S-repair for Δ is APX-complete.*

Proof. If no simplification can be applied to Δ, then, as we explained above, there are at least two local minima $X_1 \to Y_1$ and $X_2 \to Y_2$ in Δ. Note that we always remove trivial FDs from Δ before applying a simplification; thus, we can assume that Δ does not contain trivial FDs. One of the following holds:

(1) $\widehat{X_1} \cap cl_\Delta(X_2) = \emptyset$ and $\widehat{X_2} \cap cl_\Delta(X_1) = \emptyset$ (that is, Δ belongs to the first class). In this case, Lemma 4.6 and Lemma 4.15 imply that computing an optimal S-repair is APX-hard.

(2) $\widehat{X_1} \cap \widehat{X_2} \neq \emptyset$, $\widehat{X_1} \cap X_2 = \emptyset$, and $\widehat{X_2} \cap X_1 = \emptyset$ (that is, Δ belongs to the second class). In this case, Lemma 4.7 and Lemma 4.16 imply that computing an optimal S-repair is APX-hard.
(3) Either $\widehat{X_1} \cap X_2 \neq \emptyset$ or $\widehat{X_2} \cap X_1 \neq \emptyset$, but not both (that is, Δ belongs to the third class). In this case, Lemma 4.7 and Lemma 4.16 imply that computing an optimal S-repair is APX-hard.
(4) $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$ and also $(X_1 \setminus X_2) \subseteq \widehat{X_2}$ and $(X_2 \setminus X_1) \subseteq \widehat{X_1}$ (that is, Δ belongs to the fourth class). Here, Lemma 4.10 and Lemma 4.17 imply that computing an optimal S-repair is APX-hard.
(5) $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$ and also either $(X_2 \setminus X_1) \not\subseteq \widehat{X_1}$ or $(X_1 \setminus X_2) \not\subseteq \widehat{X_2}$ (that is, Δ belongs to the fifth class). Lemma 4.12 and Lemma 4.18 imply that computing an optimal S-repair is APX-hard.

Proposition 3.10 implies that computing an optimal S-repair is always in APX; thus, the problem is actually APX-complete in each one of these cases. This concludes our proof of the lemma. □

At this point, we know that computing an optimal S-repair for a set of FDs that cannot be simplified is APX-complete. Next, we discuss the cases where OSRSucceeds returns false on a set of FDs that can be simplified (but cannot be reduced to a trivial set of FDs).

4.2.4 Step 4: Applying Simplifications. The OSRSucceeds algorithm starts with a set of FDs Δ and simplifies this set using our three simplifications until it is no longer possible. We now show that whenever OSRSucceeds simplifies an FD set Δ into an FD set $\Delta - X$, there is a fact-wise reduction from $(R - X, \Delta - X)$ to (R, Δ), where R is the underlying relation schema (and $R - X$ is the result of removing the attributes in X from the relation R). Therefore, Lemma 4.4 implies that if computing an optimal S-repair is APX-hard w.r.t. $\Delta - X$, then it is also APX-hard w.r.t. Δ.

Lemma 4.20. *Let $R(A_1, \ldots, A_m)$ be a relation schema and let Δ be an FD set over R. Let $X \subseteq \{A_1, \ldots, A_m\}$ be a set of attributes. Then, there is a fact-wise reduction from $(R - X, \Delta - X)$ to (R, Δ).*

Proof. We define a fact-wise reduction $\Pi : (R - X, \Delta - X) \to (R, \Delta)$, using the constant $\odot \in$ Const. Let $\mathbf{t}$ be a tuple over $R - X$. We define Π as follows:

$$\Pi(\mathbf{t}).A_k \stackrel{\text{def}}{=} \begin{cases} \odot & A_k \in X \\ t.A_k & \text{otherwise.} \end{cases}$$

It is left to show that Π is a fact-wise reduction. To do so, we prove that Π is well defined, injective, and preserves consistency and inconsistency.

Π is well defined. This is straightforward from the definition.

Π is injective. Let $\mathbf{t}, \mathbf{t}'$ be two distinct tuples over $R - X$. Since $\mathbf{t} \neq \mathbf{t}'$, there exists an attribute A_j in $\{A_1, \ldots, A_m\} \setminus X$, such that $\mathbf{t}.A_j \neq \mathbf{t}'.A_j$. Thus, it also holds that $\Pi(\mathbf{t}).A_j \neq \Pi(\mathbf{t}').A_j$ and $\Pi(\mathbf{t}) \neq \Pi(\mathbf{t}')$.

Π preserves consistency. Let $\mathbf{t}, \mathbf{t}'$ be two distinct tuples over $R - X$. We contend that the set $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t. $\Delta - X$ if and only if the set $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t. Δ.

The "if" direction. Assume that $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta - X$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ. Since $\{\mathbf{t}, \mathbf{t}'\}$ is inconsistent w.r.t. $\Delta - X$, there exists an FD $Z \to B$ in $\Delta - X$, such that $\mathbf{t}$ and $\mathbf{t}'$ agree on all the attributes on the left-hand side of the FD, but do not agree on attribute B. The FD $(Z \cup X') \to B$ (for some $X' \subseteq X$) belongs to Δ, and since $\Pi(\mathbf{t}).A_k = t.A_k$ and $\Pi(\mathbf{t}').A_k = t'.A_k$ for each attribute $A_k \notin X$, and $\Pi(\mathbf{t}).A_k = \Pi(\mathbf{t}').A_k = \odot$ for each attribute $A_k \in X$, it holds that $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all the attributes in $Z \cup X'$, but do not agree on the attribute B; thus, $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t. Δ.

The "only if" direction. Assume that $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t $\Delta - X$. We prove that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is consistent w.r.t Δ. Let us assume, by way of contradiction, that $\{\Pi(\mathbf{t}), \Pi(\mathbf{t}')\}$ is inconsistent w.r.t Δ. That is, there exists an FD $Z \rightarrow B$ in Δ, such that $\Pi(\mathbf{t})$ and $\Pi(\mathbf{t}')$ agree on all the attributes in Z, but do not agree on attribute B. Clearly, it holds that $B \notin X$ (since $\Pi(\mathbf{t}).A_i = \Pi(\mathbf{t}').A_i = \odot$ for each attribute $A_i \in X$). Note that the FD $(Z \setminus X) \rightarrow B$ belongs to $\Delta - X$. Since $\Pi(\mathbf{t}).A_k = t.A_k$ and $\Pi(\mathbf{t}').A_k = t'.A_k$ for each attribute $A_k \notin X$, the tuples $\mathbf{t}$ and $\mathbf{t}'$ also agree on all the attributes on the left-hand side of the FD $(Z \setminus X) \rightarrow B$, but they do not agree on the attribute B on its right-hand side. Thus, $\mathbf{t}$ and $\mathbf{t}'$ violate an FD in Δ, which is a contradiction to the fact that the set $\{\mathbf{t}, \mathbf{t}'\}$ is consistent w.r.t. $\Delta - X$. □

The following lemmas are straightforward based on Lemma 4.20.

Lemma 4.21. *Let R be a relation schema and let Δ be an FD set over R. If Δ has a common lhs A, then there is a fact-wise reduction from $(R - A, \Delta - A)$ to (R, Δ).*

Lemma 4.22. *Let R be a relation schema and let Δ be an FD set over R. If Δ has a consensus FD, $\emptyset \rightarrow A$, then there is a fact-wise reduction from $(R - A, \Delta - A)$ to (R, Δ).*

Lemma 4.23. *Let R be a relation schema and let Δ be an FD set over R. If Δ has an lhs marriage, (X_1, X_2), then there is a fact-wise reduction from $(R - X_1X_2, \Delta - X_1X_2)$ to (R, Δ).*

Note that we build the fact-wise reductions from $(R - X, \Delta - X)$ to (R, Δ), and not from $(R, \Delta - X)$ to (R, Δ), although the OSRSucceeds algorithm does not change the relation R. Since we remove the attributes of X from all the FDs in Δ to obtain $\Delta - X$, these attributes can be ignored when computing an optimal S-repair of a table T over R w.r.t. $\Delta - X$. This holds, since two tuples t_1 and t_2 in T are in conflict w.r.t. $\Delta - X$ if and only if the tuples t_1' and t_2', obtained by removing the attributes of X from the tuples t_1 and t_2, are in conflict w.r.t. Δ. Hence, removing the attributes of X from R does not affect the complexity of the problem.

Next, we use all the above results to prove the negative side of Theorem 3.4.

4.2.5 Combining Everything. We now prove that if OSRSucceeds(Δ) returns false, then computing an optimal S-repair is APX-complete. We prove that by induction on n, the number of simplifications that will be applied to Δ by OSRSucceeds. The basis of the induction (that is, $n = 0$), is a direct consequence of Lemma 4.19. We finish our proof with the following lemma.

Lemma 4.24. *Let R be a relation schema and let Δ be an FD set over R. If* OSRSucceeds(Δ) *returns false, then computing an optimal S-repair for Δ is APX-complete.*

Proof. We will prove the lemma by induction on n, the number of simplifications that will be applied to Δ by OSRSucceeds. The basis of the induction is $n = 0$. In this case, Lemma 4.19 implies that computing an optimal S-repair is indeed APX-complete. For the inductive step, we need to prove that if the claim is true for all $n = 1, \ldots, k - 1$, it is also true for $n = k$. In this case, OSRSucceeds(Δ) will start by applying a simplification to the problem. One of the following holds:

(1) Δ has a common lhs A. Note that OSRSucceeds(Δ) will return false only if OSRSucceeds($\Delta - A$) returns false. From the inductive step, we know that if OSRSucceeds($\Delta - A$) returns false, then computing an optimal S-repair for $\Delta - A$ is APX-complete. Thus, Lemma 4.21 implies that computing an optimal S-repair for Δ is APX-hard.
(2) Δ has a consensus FD $\emptyset \rightarrow A$. The algorithm OSRSucceeds(Δ) will return false only if OSRSucceeds($\Delta - A$) returns false. From the inductive step, we know that if OSRSucceeds($\Delta - A$) returns false, then computing an optimal S-repair for $\Delta - A$ is APX-complete. Thus, Lemma 4.22 implies that computing an optimal S-repair for Δ is APX-hard.

(3) Δ has an lhs marriage (X_1, X_2). Again, OSRSucceeds(Δ) will return false only if OSRSucceeds($\Delta - X_1X_2$) returns false. From the inductive step, we know that if OSRSucceeds($\Delta - X_1X_2$) returns false, then computing an optimal S-repair for $\Delta - X_1X_2$ is APX-complete. Thus, Lemma 4.23 implies that computing an optimal S-repair for Δ is APX-hard.

Proposition 3.10 implies that computing an optimal S-repair is always in APX; thus, the problem is APX-complete in each one of these cases. □

This concludes our proof of Theorem 3.4.

5 COMPUTING AN OPTIMAL U-REPAIR

In this section, we focus on the problem of computing an optimal U-repair and an approximation thereof. We give general results that assist in the complexity analysis of the problem, compare it to the problem of finding an optimal S-repair (discussed in the previous section), and identify sufficient conditions for efficient reductions between the two problems. Unlike S-repairs, the existence of a dichotomy for optimal U-repairs remains an open problem.

5.1 Reductions between FD Sets

For a set Δ of FDs, we use $attr(\Delta)$ to denote the set of attributes that appear in Δ (i.e., the union of lhs and rhs over all the FDs in Δ). Two FD sets Δ_1 and Δ_2 (over the same schema) are *attribute disjoint* if $attr(\Delta_1)$ and $attr(\Delta_2)$ are disjoint. For example, $\{A \to BC, C \to D\}$ and $\{E \to FG\}$ are attribute disjoint. The following proposition shows a decomposability property on the optimal U-repairs of an FD set that can be divided into two attribute-disjoint FD sets.

Proposition 5.1. *Let T be a table over a relation schema R. Let Δ be a set of FDs over R such that $\Delta = \Delta_1 \cup \Delta_2$ for two attribute disjoint FD sets Δ_1 and Δ_2. If U, U_1, U_2 are optimal U-repairs of T w.r.t. $\Delta, \Delta_1, \Delta_2$, respectively, then $dist_{\mathsf{upd}}(U, T) = dist_{\mathsf{upd}}(U_1, T) + dist_{\mathsf{upd}}(U_2, T)$.*

Proof. Let us denote the Hamming distance of a tuple $i \in ids(T)$ w.r.t. a subset of attributes $P \subseteq attr(\Delta)$ as $H_P(T[i], U[i])$ (i.e., the number of attributes $A \in P$ for which $T[i].A \neq U[i].A$). Since U is an optimal U-repair of T w.r.t. Δ, no attribute A such that $A \notin attr(\Delta)$ is updated in U. If this is not the case, then we can build another U-repair U' that agrees with U on every value in an attribute $A \in attr(\Delta)$, and agrees with T on every value in an attribute $B \notin attr(\Delta)$. The table U' also satisfies Δ and $dist_{\mathsf{upd}}(U', T) < dist_{\mathsf{upd}}(U, T)$, which is a contradiction to the fact that U is an optimal U-repair. Clearly, for each $i \in ids(T)$ it holds that $H_{attr(\Delta)}(T[i], U[i]) = H_{attr(\Delta_1)}(T[i], U[i]) + H_{attr(\Delta_2)}(T[i], U[i])$. Hence, we can build two U-repairs U_1', U_2' from U, one for Δ_1 and one for Δ_2, by taking the values of the attributes in $attr(\Delta_1)$ (respectively, $attr(\Delta_2)$) from U, and keeping the remaining values unchanged. Then, it holds that $dist_{\mathsf{upd}}(U, T) = dist_{\mathsf{upd}}(U_1', T) + dist_{\mathsf{upd}}(U_2', T)$. It is only left to prove that U_1' and U_2' are optimal U-repairs of T w.r.t. Δ_1 and Δ_2, respectively.

Let us assume, by way of contradiction, that U_1' is not an optimal U-repair of T w.r.t. Δ_1 (a similar proof applies for U_2'). Then, we can construct another U-repair U' of T w.r.t. Δ, by taking the values of the attributes in $attr(\Delta_1)$ from U_1 (which is an optimal U-repair of T w.r.t. Δ_1), the values of the attributes in $attr(\Delta_2)$ from U_2', and the rest of the values from the original table T. Since U_1 is an optimal U-repair of T w.r.t. Δ_1, it does not update any attribute that does not appear in $attr(\Delta_1)$. Thus, it again holds that $dist_{\mathsf{upd}}(U', T) = dist_{\mathsf{upd}}(U_1, T) + dist_{\mathsf{upd}}(U_2', T)$, and since $dist_{\mathsf{upd}}(U_1, T) < dist_{\mathsf{upd}}(U_1', T)$, it holds that $dist_{\mathsf{upd}}(U', T) < dist_{\mathsf{upd}}(U, T)$, which is a contradiction to the fact that U is an optimal U-repair of T w.r.t. Δ. □

Using Proposition 5.1, we can show the following theorem, which implies that to determine the complexity of the union of two attribute-disjoint FD sets, it suffices to look at each set separately.

Theorem 5.2. *Suppose that $\Delta = \Delta_1 \cup \Delta_2$ where Δ_1 and Δ_2 are attribute disjoint. The following hold.*

(1) There is a polynomial-time algorithm that, given a table T and α-optimal U-repairs of T w.r.t. Δ_1 and Δ_2, produces an α-optimal U-repair of T w.r.t. Δ.
(2) For each $i \in \{1, 2\}$, there is a strict reduction from the problem of computing an α-optimal U-repair for Δ_i to the problem of computing an α-optimal U-repair for Δ.

Proof. (1) We argue that an α-optimal U-repair U^α of T w.r.t. Δ can be obtained by composing α-optimal U-repairs U_1^α, U_2^α of T w.r.t. Δ_1, Δ_2, respectively. Suppose that U, U_1, U_2 are optimal U-repairs of T w.r.t. $\Delta, \Delta_1, \Delta_2$, respectively. Since U_1^α, U_2^α are α-optimal U-repairs, the following holds:

$$dist_{\text{upd}}(U_1^\alpha, T) \leq \alpha\, dist_{\text{upd}}(U_1, T) \text{ and } dist_{\text{upd}}(U_2^\alpha, T) \leq \alpha\, dist_{\text{upd}}(U_2, T).$$

We construct a U-repair U^α of T w.r.t. Δ by copying the values of the attributes in $attr(\Delta_1)$ from U_1 and the values of the attributes in $attr(\Delta_2)$ from U_2. The rest of the values are copied from the original table T. The following holds:

$$\begin{aligned} dist_{\text{upd}}(U^\alpha, T) &\leq dist_{\text{upd}}(U_1^\alpha, T) + dist_{\text{upd}}(U_2^\alpha, T) \leq \alpha(dist_{\text{upd}}(U_1, T) + dist_{\text{upd}}(U_2, T)) \\ &= \alpha\, dist_{\text{upd}}(U, T) \quad \text{(from Proposition 5.1)}. \end{aligned}$$

Hence, U^α is an α-optimal U-repair of T w.r.t. Δ.

(2) We prove the claim for Δ_1. The proof for Δ_2 is symmetric. Given an input T to the first problem, we construct an input $f(T)$ (that we denote by T_0) to the second problem by updating the values of all the attributes A such that $A \notin attr(\Delta_1)$ to $\mathtt{0}$, and keeping the rest of the values unchanged. Let U, U_1, U_2 denote optimal U-repairs of T_0 w.r.t. $\Delta, \Delta_1, \Delta_2$, respectively. Proposition 5.1 implies that

$$dist_{\text{upd}}(U, T_0) = dist_{\text{upd}}(U_1, T_0) + dist_{\text{upd}}(U_2, T_0).$$

No FDs in Δ_2 are violated by the tuples in T_0, since Δ_1 and Δ_2 are attribute disjoint, and it holds that $T_0[i].A = \mathtt{0}$ for every tuple $i \in ids(T_0)$ and every attribute $A \notin attr(\Delta_1)$. Hence, $U_2 = T_0$, and $dist_{\text{upd}}(U_2, T_0) = 0$. Therefore,

$$dist_{\text{upd}}(U, T_0) = dist_{\text{upd}}(U_1, T_0).$$

Now, given an α-optimal U-repair U^α of T_0 w.r.t. Δ, we generate a table U_1^α by copying the values of all the attributes in $attr(\Delta_1)$ from U^α and keeping the other attributes unchanged (i.e., as in T_0). Since U^α is an α-optimal U-repair of T_0 w.r.t. Δ, the following holds:

$$dist_{\text{upd}}(U^\alpha, T_0) \leq \alpha\, dist_{\text{upd}}(U, T_0).$$

Then, we have $dist_{\text{upd}}(U_1^\alpha, T_0) \leq dist_{\text{upd}}(U^\alpha, T_0) \leq \alpha\, dist_{\text{upd}}(U, T_0) = \alpha\, dist_{\text{upd}}(U_1, T_0)$.

Since the attributes that do not belong to $attr(\Delta_1)$ are unchanged in U_1^α and U_1, and the values of the attributes in $attr(\Delta_1)$ are the same in T_0 and T, we can construct an α-optimal U-repair U'^α_1 of T from U_1^α by copying all the values of the attributes in $attr(\Delta_1)$ from U_1^α and copying the rest of the values from T. Similarly, we can construct an optimal U-repair U_1' of T from U_1 by copying all the values of the attributes in $attr(\Delta_1)$ from U_1 and copying the rest of the values from T. Clearly, it holds that $dist_{\text{upd}}(U_1^\alpha, T_0) = dist_{\text{upd}}(U'^\alpha_1, T)$ and $dist_{\text{upd}}(U_1, T_0) = dist_{\text{upd}}(U_1', T)$. Thus,

$$dist_{\text{upd}}(U'^\alpha_1, T) \leq \alpha\, dist_{\text{upd}}(U_1', T).$$

This gives us an α-optimal U-repair of T w.r.t. Δ_1 and that concludes our proof. □

Example 5.3. Consider the following set of FDs:

$$\Delta \stackrel{\text{def}}{=} \{\mathsf{item} \to \mathsf{cost}\,,\ \mathsf{buyer} \to \mathsf{address}\}.$$

We will later show that if Δ consists of a single FD, then an optimal U-repair can be computed in polynomial time. Hence, we can compute an optimal U-repair under $\Delta_1 = \{\text{item} \rightarrow \text{cost}\}$ and under $\Delta_2 = \{\text{buyer} \rightarrow \text{phone}\}$. Then, Theorem 5.2 implies that an optimal U-repair can be computed in polynomial time under Δ as well.

Now consider the following set of FDs:

$$\Delta' \stackrel{\text{def}}{=} \{\text{item} \rightarrow \text{cost}\,,\ \text{buyer} \rightarrow \text{address}\,,\ \text{address} \rightarrow \text{state}\}.$$

Kolahi and Lakshmanan [33] proved that it is NP-hard to compute an optimal U-repair for $\{A \rightarrow B, B \rightarrow C\}$, by reduction from the problem of finding a minimum vertex cover of a graph G. Their reduction is, in fact, a PTAS reduction if we use vertex cover in a graph of a bounded degree [2]. Hence, computing an optimal U-repair is APX-hard for this set of FDs. Theorem 5.2 then implies that it is also APX-hard for Δ'.

5.2 Reductions by Removing Consensus FDs

Next, we discuss the problem in the presence of consensus FDs. The first property we show below states that the problem of computing an optimal U-repair w.r.t. a consensus FD can be solved in polynomial time.

Proposition 5.4. *An optimal U-repair for $\Delta = \{\emptyset \rightarrow A\}$ is computable in polynomial time.*

Proof. Consider a table T over a schema R that violates the FD $\emptyset \rightarrow A$. That is, at least two tuples in T disagree on the value of attribute A. We obtain a U-repair U of T in the following way. For every value a that appears in attribute A, we compute the total weight W_a of the tuples in $\sigma_{A=a}T$ (i.e., $W_a = w_T(\sigma_{A=a}T)$). Then, we choose the value a_0, having the maximum weight W_a among all such a's (that is, $W_{a_0} \geq W_a$ for all $a \in \pi_A T[*]$). We keep the tuples in $\sigma_{A=a_0}T$ unchanged, and update the value of attribute A in every other tuple $\mathbf{t} \in \sigma_{A=a}T$ for $a \neq a_0$ to a_0. Clearly, U is consistent, since now every tuple $\mathbf{t}$ in T will have $\mathbf{t}.A = a_0$.

To see that U is an optimal U-repair, first note that a repair with a lower distance cannot be obtained by setting the A values to a fresh constant from the infinite domain, since choosing a value from the active domain saves us the cost of the repair for at least one tuple in T. Now, assume that some other value $a_1 \neq a_0$ has been chosen for all tuples in T in a repair U_1 for which $dist_{\text{upd}}(U_1, T) < dist_{\text{upd}}(U, T)$. Note that $H(T[i], U_1[i]) = 0$ if $i \in ids(\sigma_{A=a_1}T)$ and $H(T[i], U_1[i]) = 1$ otherwise. Thus,

$$dist_{\text{upd}}(U_1, T) = \sum_{i \in ids(T)} w_T(i) \cdot H(T[i], U_1[i]) = \sum_{a \neq a_1} \sum_{i \in ids(\sigma_{A=a}T)} w_T(i) = \sum_{a \neq a_1} W_a = \sum_{i \in ids(T)} w_T(i) - W_{a_1},$$

whereas

$$dist_{\text{upd}}(U, T) = \sum_{i \in ids(T)} w_T(i) - W_{a_0}.$$

Since $W_{a_0} \geq W_{a_1}$, it holds that $dist_{\text{upd}}(U_1, T) \geq dist_{\text{upd}}(U, T)$, which is a contradiction the assumption that $dist_{\text{upd}}(U_1, T) < dist_{\text{upd}}(U, T)$. Hence, U is an optimal U-repair. □

Recall that, for a set Δ of FDs and a set X of attributes, the set $\Delta - X$ denotes the set of FDs that is obtained from Δ by removing each attribute of X from the lhs and rhs of every FD. Also recall that $cl_\Delta(\emptyset)$ is the set of all consensus attributes. Using the above results, we prove Theorem 5.5, which states that consensus FDs do not change the complexity of the problem. The proof uses Theorem 5.2, and a special treatment of the case where all FDs are consensus FDs using Proposition 5.4.

Theorem 5.5. *Let Δ be a set of FDs. There is a strict reduction from the problem of computing an optimal U-repair for Δ to that of computing an optimal U-repair for $\Delta - cl_\Delta(\emptyset)$, and vice versa.*

Proof. We start by proving that there is a strict reduction from the problem of computing an optimal U-repair for Δ to that of computing an optimal U-repair for $\Delta - cl_\Delta(\emptyset)$. Proposition 5.4 states that an optimal U-repair w.r.t. $\{\emptyset \rightarrow cl_\Delta(\emptyset)\}$ (which is also an α-optimal U-repair for any $\alpha \geq 1$) can be computed in polynomial time. Therefore, using the fact that Δ and $\{\emptyset \rightarrow cl_\Delta(\emptyset)\} \cup (\Delta - cl_\Delta(\emptyset))$ are equivalent and $\{\emptyset \rightarrow cl_\Delta(\emptyset)\}$ and $(\Delta - cl_\Delta(\emptyset))$ are attribute disjoint, this direction follows from the first part of Theorem 5.2. The second direction follows immediately from the second part of Theorem 5.2. □

As an example of applying Theorem 5.5, if Δ consists of *only* consensus FDs, then an optimal U-repair can be computed in polynomial time, since $\Delta - cl_\Delta(\emptyset)$ is empty. As another example, if Δ is the set $\{\emptyset \rightarrow D, AD \rightarrow B, B \rightarrow CD\}$, then $\Delta - cl_\Delta(\emptyset) = \{A \rightarrow B, B \rightarrow C\}$ and, according to Theorem 5.5, computing an optimal U-repair is APX-hard, since this problem is hard for $\{A \rightarrow B, B \rightarrow C\}$ due to Kolahi and Lakshmanan [33], as explained in Example 5.3.

5.3 Reductions to/from Subset Repairing

In this section, we establish several results that enable us to infer complexity results for the problem of computing an optimal U-repair from that of computing an optimal S-repair via polynomial-time reductions. We first need a notation.

Let Δ be a set of FDs. An *lhs cover* of Δ is a set C of attributes that hits every nonempty lhs, that is, $X \cap C \neq \emptyset$ for every $X \rightarrow Y$ in Δ, such that $X \neq \emptyset$. We denote the minimum cardinality of an lhs cover of Δ by $mlc(\Delta)$. For instance, if Δ is nonempty and has a common lhs (e.g., Figure 1), then $mlc(\Delta) = 1$.

The results of this section are based on the following proposition, which shows that we can transform a consistent update into a consistent subset (with no extra cost) and, in the absence of consensus FDs, a consistent subset into a consistent update (with some extra cost). We give the proof here, as it shows the actual constructions.

Proposition 5.6. *Let Δ be a set of FDs and T a table. The following can be done in polynomial time.*

(1) Given a consistent update U, construct a consistent subset S such that $dist_{\mathsf{sub}}(S, T) \leq dist_{\mathsf{upd}}(U, T)$.
(2) Given a consistent subset S, and assuming that Δ is consensus free, construct a consistent update U such that $dist_{\mathsf{upd}}(U, T) \leq mlc(\Delta) \cdot dist_{\mathsf{sub}}(S, T)$.

Proof. For (1), we construct S from U by excluding any $i \in ids(T)$ such that $T[i]$ has at least one attribute updated in U (i.e., $H(T[i], U[i]) \geq 1$). For (2), we construct U from S as follows. Let C be an lhs cover of minimum cardinality $mlc(\Delta)$. The tuple of each $i \in ids(S)$ is left intact, and for $i \in ids(T) \setminus ids(S)$, we update the value of $T[i].A$ for each attribute $A \in C$ to a fresh constant from our infinite domain Val. Since C is an lhs cover and there are no consensus FDs, for all $X \rightarrow Y$ in Δ it holds that two distinct tuples in U that agree on X must correspond to intact tuples; hence, U is consistent (as S is consistent). □

As we discuss later in Section 5.4, Proposition 5.6, combined with Proposition 3.10, reestablishes the result of Kolahi and Lakshmanan [33], stating that computing a U-repair is in APX. We also establish the following additional consequences of Proposition 5.6. The first is an immediate corollary (that we refer to later on) about the relationship between optimal repairs.

Corollary 5.7. *Let Δ be a set of FDs, T a table, S an optimal S-repair of T, and U an optimal U-repair of T. Then, $dist_{\text{sub}}(S,T) \leq dist_{\text{upd}}(U,T)$. Moreover, if Δ is consensus free, then $dist_{\text{upd}}(U,T) \leq mlc(\Delta) \cdot dist_{\text{sub}}(S,T)$.*

The second consequence relates to FD sets Δ with a common lhs, that is, $mlc(\Delta) = 1$.

Corollary 5.8. *Let Δ be an FD set with a common lhs. There is a strict reduction from the problem of computing an optimal S-repair to that of computing an optimal U-repair, and vice versa.*

For example, if Δ consists of a single FD, then an optimal U-repair can be computed in polynomial time. Additional examples follow.

Example 5.9. To illustrate the use of Corollary 5.8, consider the FD set Δ of our running example (Figure 1). Since Δ has a common lhs, and we have established in Example 3.5 that an optimal S-repair for Δ can be found in polynomial time (i.e., Δ passes the test of OSRSucceeds), we get that an optimal U-repair can also be computed in polynomial time for Δ.

As another illustration, consider the following FD set:

$$\Delta_1 \stackrel{\text{def}}{=} \{\text{id country} \rightarrow \text{passport}\,,\ \text{id passport} \rightarrow \text{country}\}.$$

Again, Δ_1 has a common lhs and Δ_1 passes the test of OSRSucceeds (by applying common lhs followed by an lhs marriage), and therefore, Theorem 3.4 implies that an optimal U-repair can be found in polynomial time.

Finally, consider the following set of FDs:

$$\Delta_2 \stackrel{\text{def}}{=} \{\text{state city} \rightarrow \text{zip}\,,\ \text{state zip} \rightarrow \text{country}\}.$$

The reader can verify that Δ_2 fails OSRSucceeds, and therefore, from Theorem 3.4, we conclude that computing an optimal U-repair is APX-complete.

By combining Theorem 5.5, Corollary 5.8, and Corollary 3.6, we conclude the following.

Corollary 5.10. *If Δ is a chain FD set, then an optimal U-repair is computable in polynomial time.*

Proof. If Δ is a chain FD set, then so is $\Delta - cl_\Delta(\emptyset)$. Theorem 5.5 states that computing an optimal U-repair has the same complexity under the two FD sets. Moreover, if $\Delta - cl_\Delta(\emptyset)$ is nonempty, then it has at least one common lhs. From Corollary 5.8, we conclude that the problem then strictly reduces to computing an S-repair, which, by Corollary 3.6, can be done in polynomial time. □

Hence, for chain FD sets, an optimal repair can be computed for both subset and update variants.

Comparison to S-Repairs. Corollaries 5.8 and 5.10 state cases where computing an optimal S-repair has the same complexity as computing an optimal U-repair. A basic case (among others) that the corollaries do not cover is in Proposition 5.11, which uses $\Delta = \{A \rightarrow B, B \rightarrow A\}$, and where again both variants have the same (polynomial-time) complexity. In the proof of Proposition 5.11, we show that, even though $mlc(\Delta)$ is 2, we have $dist_{\text{upd}}(U,T) = dist_{\text{sub}}(S,T)$ for all tables T over R, optimal U-repairs U and optimal S-repairs S. Since Δ passes the test of OSRSucceeds (by applying lhs marriage), from Theorem 3.4 an optimal S-repair can be computed in polynomial time, and therefore an optimal U-repair of $\{A \rightarrow B, B \rightarrow A\}$ can also be computed in polynomial time (note that the FDs $\{A \rightarrow B, B \rightarrow A\}$ imply that in a consistent update, one value of A cannot be associated with multiple values of B and vice versa).

Proposition 5.11. *An optimal U-repair for $\Delta = \{A \rightarrow B, B \rightarrow A\}$ can be computed in polynomial time.*

Proof. Let T be a table over a relation schema R. We now prove that $dist_{\mathsf{upd}}(U, T) = dist_{\mathsf{sub}}(S, T)$ for an optimal U-repair U and an optimal S-repair S of T w.r.t. Δ. Corollary 5.7 implies that

$$dist_{\mathsf{sub}}(S, T) \leq dist_{\mathsf{upd}}(U, T). \tag{4}$$

Given S, we can construct a consistent update U of T by keeping the tuples in S unchanged, and updating the value in either attribute A or attribute B for the rest of the tuples, as follows. Consider any tuple $\mathbf{t} \in T \setminus S$. There must exist a tuple $\mathbf{t}' \in S$ with either $\mathbf{t}.A = \mathbf{t}'.A$ or $\mathbf{t}.B = \mathbf{t}'.B$; otherwise, $\mathbf{t}$ could be included in S, which contradicts the optimality of S. If there exists a tuple $\mathbf{t}' \in S$ with $\mathbf{t}.A = \mathbf{t}'.A$, then we change the value of $\mathbf{t}.B$ to the value of $\mathbf{t}'.B$. Analogously, if there exists a tuple $\mathbf{t}' \in S$ with $\mathbf{t}.B = \mathbf{t}'.B$, then we change the value of $\mathbf{t}.A$ to the value of $\mathbf{t}'.A$. Hence, we get a consistent update U with

$$dist_{\mathsf{sub}}(S, T) \geq dist_{\mathsf{upd}}(U, T). \tag{5}$$

Combining the inequalities Equations (4) and (5), we get that U is an optimal U-repair of T w.r.t. Δ.

Since Δ passes the test of OSRSucceds, Theorem 3.4 implies that an optimal S-repair S can be computed in polynomial time, from which an optimal U-repair can be computed in polynomial time as described above. □

Do the two variants of optimal repairs feature the same complexity for every *set of FDs?* Next, we answer this question in a negative way.

We have already seen an example of an FD set Δ where an optimal U-repair can be computed in polynomial time, but finding an S-repair is APX-complete. Indeed, Example 5.3 shows that $\{A \rightarrow B, C \rightarrow D\}$ is a tractable case for optimal U-repairs; yet, it fails the test of OSRSucceeds, and is therefore hard for optimal S-repairs (Theorem 3.4). The question of whether there exists a set of FDs for which computing an optimal U-repair is hard, while computing an optimal S-repair can be done in polynomial time remains open.[4]

5.4 Approximation

In this section, we discuss approximations for optimal U-repairs. We restrict the discussion to FD sets Δ that are nonempty and consensus free. Note that this is not a limiting assumption, since empty Δs are trivially tractable, and consensus FDs can be eliminated, without increasing the approximation ratio, due to Theorem 5.5.

The combination of Propositions 3.10 and 5.6 gives the following.

Theorem 5.12. *An α-optimal U-repair can be computed in polynomial time for $\alpha = 2 \cdot mlc(\Delta)$.*

Observe that the approximation ratio can be further improved by applying Theorem 5.2: if Δ is the union of attribute-disjoint FD sets Δ_1 and Δ_2, then an α-optimal U-repair can be computed (under Δ) where $\alpha = 2 \cdot \max\{mlc(\Delta_1), mlc(\Delta_2)\}$.

Kolahi and Lakshmanan [33] gave a constant-factor approximation algorithm for U-repairs (assuming Δ is fixed). We first explain their ratio, and then compare it to ours.

Let Δ be a set of FDs, and assume (without loss of generality) that the rhs of each FD consists of a single attribute. By $MFS(\Delta)$, we denote the maximum number of attributes in the lhs of any FD in Δ. An *implicant* of an attribute A is a set X of attributes such that $X \rightarrow A$ is entailed by Δ. A *core implicant* of A is a set C of attributes that hits every implicant of A (i.e., $X \cap C \neq \emptyset$ whenever $X \rightarrow A$ is in the closure of Δ). A *minimum core implicant* of A is a core implicant of A with the

[4] In the conference paper Reference [37], we claimed that $\{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$ is an example; however, we later found an error in our proof of hardness for U-repairs. In fact, the complexity of computing an optimal U-repair is still open for this case.

smallest cardinality. By $MCI(\Delta)$, we denote the size of the largest minimum core implicant over all attributes A.

Theorem 5.13. *[33] An α-optimal U-repair can be computed in polynomial time where $\alpha = (MCI(\Delta) + 2) \cdot (2MFS(\Delta) - 1)$.*

In both Theorems 5.12 and 5.13, the approximation ratios are constants under data complexity, but depend on Δ. It is still unknown whether there is a constant α that applies to all FD sets Δ. Yet, it is known that a constant-ratio approximation cannot be obtained in polynomial time under *combined complexity* (where R, T, and Δ are all given as input) [33].

Although the proof of Theorem 5.12 is much simpler than the non-trivial proof of Theorem 5.13 given in Reference [33], it can be noted that the approximation ratios in these two theorems are not directly comparable. If k is the number of attributes, then the worst-case approximation ratio in Theorem 5.13 is quadratic in k, while the worst-case approximation in Theorem 5.12 is linear in k (precisely, linear in $\min(k, |\Delta|)$). Moreover, an easy observation is that the ratio between the two approximation ratios can be at most linear in k. In the remainder of this section, we illustrate the difference between the approximations with examples.

First, we show an infinite sequence of FD sets where the approximation ratio of Theorem 5.12 is $\Theta(k)$ and that of Theorem 5.13 is $\Theta(k^2)$. For a natural number $k \geq 1$, we define Δ_k as follows:

$$\Delta_k \stackrel{\text{def}}{=} \{A_0 \cdots A_k \rightarrow B_0, B_0 \rightarrow C, B_1 \rightarrow A_0, \ \ldots, \ B_k \rightarrow A_0\}.$$

The approximation ratio for Δ_k given by Theorem 5.12 is $2(k + 2)$. For the approximation ratio of Theorem 5.13, we have $MFS(\Delta_k) = k + 1$ (due to the FD $A_0 \cdots A_k \rightarrow B$) and $MCI(\Delta_k) = k$ (since the core implicant of A_0 is $\{B_1, \ldots, B_k\}$). Hence, the approximation ratio of Theorem 5.13 grows quadratically with k (i.e., it is $\Theta(k^2)$).

However, below is a sequence of FD sets in which the approximation ratio of Theorem 5.12 grows linearly with k, while that of Theorem 5.13 is a constant:

$$\Delta'_k \stackrel{\text{def}}{=} \{A_0A_1 \rightarrow B_0, \ A_1A_2 \rightarrow B_1, \ \ldots, \ A_kA_{k+1} \rightarrow B_k\}.$$

Here, it holds that $mlc(\Delta'_k) = \lceil (k + 1)/2 \rceil$, that $MFS(\Delta'_k) = 2$, and that $MCI(\Delta'_k) = 1$. Therefore, the approximation ratio of Theorem 5.12 is $\Theta(k)$ while that of Theorem 5.13 is constant.

The following two theorems show that computing an optimal U-repair is hard for both Δ_k and Δ'_k; thus, an approximation is, indeed, needed. The proofs of both theorems are in the Appendix.

Theorem 5.14. *Let $k \geq 1$ be fixed. Then, computing an optimal U-repair is APX-complete for $R(A_0, \ldots, A_k, B_0, \ldots, B_k, C)$ and Δ_k.*

Theorem 5.15. *Let $k \geq 1$ be fixed. Then, computing an optimal U-repair is APX-complete for $R(A_0, \ldots, A_{k+1}, B_0, \ldots, B_k)$ and Δ'_k.*

Clearly, one can take the benefit of the approximations of both Theorems 5.12 and 5.13 by computing U-repairs by both algorithms and selecting the one with the smaller cost. As we showed, this combined approximation outperforms each of its two components.

6 DISCUSSION AND FUTURE WORK

We investigated the complexity of computing an optimal S-repair and an optimal U-repair. For the former, we established a dichotomy over all sets of FDs (and schemas). For the latter, we developed general techniques for complexity analysis, showed concrete complexity results, and explored the connection to the complexity of S-repairs. We presented approximation results and, in the case of U-repairs, compared to the approximation of Kolahi and Lakshmanan [33]. In the case of S-repairs, we drew a direct connection to probabilistic database repairs, and completed a dichotomy

by Gribkoff et al. [28] to the entire space of FDs. Quite a few directions are left for future investigation, and we conclude with a discussion of some of these.

Some immediate open problems remain for future investigation. For one, our understanding of the complexity of computing an optimal U-repair is considerably more restricted than that of an optimal S-repair. We would like to complete our complexity analysis for optimal U-repairs into a full dichotomy. In particular, it remains open whether there is any set of FDs such that computing an optimal U-repair is hard while computing an optimal S-repair is tractable. More fundamentally, we would like to incorporate restrictions on the allowed value updates. Our results are heavily based on the ability to update *any cell* with *any value* from an infinite domain. A natural restriction on the update repairs is to allow revising only certain attributes, possibly using a finite (or even small) space of possible new values. It is not clear how to incorporate such a restriction in our results and proof techniques.

As our results are restricted to FDs, an obvious important direction is to extend our study to other types of integrity constraints, such as denial constraints [25], conditional FDs [13], referential constraints [21], and tuple-generating dependencies [9]. Moreover, the repair operations we considered are either exclusively tuple deletions or exclusively value updates. Hence, another clear direction is to allow mixtures of deletions, insertions and updates, where the cost depends on the operation type, the involved tuple, the involved attribute (in the case of updates), and the new and old attribute values.

In the case of S-repairs, we are interested in incorporating *preferences*, as in the framework of *prioritized repairing* by Staworko et al. [41]. There, priorities among tuples allow us to eliminate subset repairs that are inferior to others (where "inferior" has several possible interpretations). It may be the case that priorities are rich enough to clean the database unambiguously [32]. A relevant question is, then, what is the minimal number of tuples that we need to delete to have an unambiguous repair? Alternatively, how many preferences are needed for this cause?

REFERENCES

[1] Foto N. Afrati and Phokion G. Kolaitis. 2009. Repair checking in inconsistent databases: Algorithms and complexity. In *Proceedings of the ICDT*. ACM, 31–41.

[2] Paola Alimonti and Viggo Kann. 2000. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.* 237, 1–2 (2000), 123–134.

[3] Omid Amini, Stéphane Pérennes, and Ignasi Sau. 2009. Hardness and approximation of traffic grooming. *Theor. Comput. Sci.* 410, 38–40 (2009), 3751–3760.

[4] Periklis Andritsos, Ariel Fuxman, and Renée J. Miller. 2006. Clean answers over dirty databases: A probabilistic approach. In *Proceedings of the ICDE*. IEEE Computer Society, 30.

[5] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the PODS*. ACM, 68–79.

[6] Ahmad Assadi, Tova Milo, and Slava Novgorodov. 2017. DANCE: Data cleaning with constraints and experts. In *Proceedings of the ICDE*. IEEE Computer Society, 1409–1410.

[7] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. 1999. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties* (1st ed.). Springer-Verlag, Berlin.

[8] Reuven Bar-Yehuda and Shimon Even. 1981. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algor.* 2, 2 (1981), 198–203.

[9] Catriel Beeri and Moshe Y. Vardi. 1984. Formal systems for tuple and equality generating dependencies. *SIAM J. Comput.* 13, 1 (1984), 76–98.

[10] Moria Bergman, Tova Milo, Slava Novgorodov, and Wang-Chiew Tan. 2015. QOCO: A query oriented data cleaning system with oracles. *Proc. Very Large Data Base* 8, 12 (2015), 1900–1903.

[11] Leopoldo E. Bertossi. 2018. Measuring and computing database inconsistency via repairs. In *Proceedings of the SUM* (Lecture Notes in Computer Science), Vol. 11142. Springer, 368–372.

[12] Leopoldo E. Bertossi. 2018. Repair-based degrees of database inconsistency: Computation and complexity. *CoRR* abs/1809.10286 (2018).

[13] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In *Proceedings of the ICDE*. IEEE, 746–755.
[14] Douglas Burdick, Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. 2017. Expressive power of entity-linking frameworks. In *Proceedings of the ICDT (LIPIcs)*, Vol. 68. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 10:1–10:18.
[15] Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. 1984. Inclusion dependencies and their interaction with functional dependencies. *J. Comput. Syst. Sci.* 28, 1 (1984), 29–59.
[16] Jan Chomicki and Jerzy Marcinkowski. 2005. Minimal-change integrity maintenance using tuple deletions. *Info. Comput.* 197, 1–2 (2005), 90–121.
[17] E. F. Codd. 1975. Recent investigations in relational data base systems. In *Proceedings of the Conference on Data: Its Use, Organization, and Management (ACM Pacific'75)*. 15–20.
[18] P. Crescenzi. 1997. A short guide to approximation preserving reductions. In *Proceedings of the IEEE*CCC. IEEE Computer Society, Washington, DC, 262.
[19] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: A commodity data cleaning system. In *Proceedings of the SIGMOD*. ACM, 541–552.
[20] Nilesh N. Dalvi and Dan Suciu. 2004. Efficient query evaluation on probabilistic databases. In *Proceedings of the VLDB*. Morgan Kaufmann, 864–875.
[21] C. J. Date. 1981. Referential integrity. In *Proceedings of the VLDB*. VLDB Endowment, 2–12.
[22] Jianfeng Du, Guilin Qi, and Yi-Dong Shen. 2013. Weight-based consistent query answering over inconsistent SHIQ knowledge bases. *Knowl. Info. Syst.* 34, 2 (2013), 335–371.
[23] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. 2015. Dichotomies in the complexity of preferred repairs. In *Proceedings of the PODS*. ACM, 3–15.
[24] Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management*. Morgan & Claypool Publishers.
[25] Terry Gaasterland, Parke Godfrey, and Jack Minker. 1992. An overview of cooperative answering. *J. Intell. Info. Syst.* 1, 2 (1992), 123–157.
[26] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC data-cleaning framework. *Proc. Very Large Data Base* 6, 9 (2013), 625–636.
[27] John Grant and Anthony Hunter. 2017. Analysing inconsistent information using distance-based measures. *Int. J. Approx. Reason.* 89 (2017), 3–26.
[28] Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. 2014. The most probable database problem. In *Proceedings of the BUDA*.
[29] Venkatesan Guruswami. 2004. Inapproximability results for set splitting and SatisfiabilityProblems with no mixed clauses. *Algorithmica* 38, 3 (1 Mar 2004), 451–469.
[30] S. Khanna, M. Sudan, and L. Trevisan. 1997. Constraint satisfaction: The approximability of minimization problems. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*. 282–296.
[31] Benny Kimelfeld. 2012. A dichotomy in the complexity of deletion propagation with functional dependencies. In *Proceedings of the PODS*. 191–202.
[32] Benny Kimelfeld, Ester Livshits, and Liat Peterfreund. 2017. Detecting ambiguity in prioritized database repairing. In *Proceedings of the ICDT*. 17:1–17:20.
[33] Solmaz Kolahi and Laks V. S. Lakshmanan. 2009. On approximating optimum repairs for functional dependency violations. In *Proceedings of the ICDT*, Vol. 361. ACM, 53–62.
[34] Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Res. Logist. Quart.* 2, 1–2 (Mar. 1955), 83–97.
[35] Ester Livshits, Ihab F. Ilyas, Benny Kimelfeld, and Sudeepa Roy. 2019. Principles of progress indicators for database repairing. *CoRR* abs/1904.06492 (2019).
[36] Ester Livshits and Benny Kimelfeld. 2017. Counting and enumerating (preferred) database repairs. In *Proceedings of the PODS*. 289–301.
[37] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2018. Computing optimal repairs for functional dependencies. In *Proceedings of the PODS*. 225–237.
[38] Andrei Lopatenko and Leopoldo E. Bertossi. 2007. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *Proceedings of the ICDT*. 179–193.
[39] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic data repairs with probabilistic inference. *Proc. Very Large Data Base* 10, 11 (2017), 1190–1201.
[40] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. 2019. A formal framework for probabilistic unclean databases. In *Proceedings of the ICDT*. 6:1–6:18.
[41] Slawek Staworko, Jan Chomicki, and Jerzy Marcinkowski. 2012. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64, 2–3 (2012), 209–246.

[42] Dan Suciu, Dan Olteanu, R. Christopher, and Christoph Koch. 2011. *Probabilistic Databases* (1st ed.). Morgan & Claypool Publishers.
[43] Ingo Wegener and R. Pruim. 2005. *Complexity Theory: Exploring the Limits of Efficient Algorithms*. Springer-Verlag, Berlin.

Received November 2018; revised May 2019; accepted August 2019