The VLDB Journal
The International Journal on
Very Large Data Bases

ISSN 1066-8888

The VLDB Journal
DOI 10.1007/
s00778-011-0239-5

A 42722
Volume 13 · Number 3 · September 2004
The VLDB Journal
13/3
2004
The International Journal on Very Large Data Bases
acm PRESS
Association for Computing Machinery
Springer

The VLDB Journal
DOI 10.1007/s00778-011-0239-5

REGULAR PAPER

Design by example for SQL table definitions with functional dependencies

Sven Hartmann · Markus Kirchberg · Sebastian Link

Received: 24 June 2010 / Revised: 25 May 2011 / Accepted: 10 June 2011

Abstract A database is C-Armstrong for a given set of constraints in a class C if it satisfies every constraint of the set and violates every constraint in C not implied by the set. Therefore, Armstrong databases are test data that perfectly illustrate the current perceptions about the semantics of a schema. We extend the existing theory of Armstrong relations to a toolbox of Armstrong tables. That is, we investigate structural and computational properties of Armstrong tables for the class of functional dependencies (FDs) over SQL tables. Relations are special instances of SQL tables with no duplicate rows and no null value occurrences. While FDs do not enjoy Armstrong tables, the combined class of standard FDs and NOT NULL constraints does enjoy Armstrong tables. The problem of finding an Armstrong table is shown to be precisely exponential for this combined class. However, we establish an algorithm that computes Armstrong tables with a size at most quadratic in that of a minimum-sized Armstrong table. Our resulting toolbox of Armstrong tables can be applied by data engineers to concisely visualize constraints on SQL data. Such support can lead to designs that guarantee efficient data management in practice.

Electronic supplementary material The online version of this article (doi:10.1007/s00778-011-0239-5) contains supplementary material, which is available to authorized users.

S. Hartmann
Department of Informatics, Clausthal University of Technology, Clausthal-Zellerfeld, Germany

M. Kirchberg
Services Platform Lab, Hewlett-Packard Laboratories, Singapore, Singapore

S. Link (✉)
School of Information Management, Victoria University of Wellington, Wellington, New Zealand
e-mail: sebastian.link@vuw.ac.nz

Keywords Armstrong table · Functional dependency · Schema design · SQL · Test data · Uniqueness

1 Introduction

A database system is a software package that manages a collection of persistent information in a shared, reliable, effective, and efficient way. The core of most database systems is still founded on *the relational model of data* [28]. In this model, data are stored in a collection of relations that may vary over time. A relation is a set of tuples over a given time-invariant relation schema.

Relations permit the storage of inconsistent data, i.e., data that violate conditions which every meaningful relation ought to satisfy. Consequently, data engineers must identify the conditions, called data dependencies, that restrict the relations to those that are meaningful for the application domain at hand. Functional dependencies (FDs) constitute the probably most important class of data dependencies [1,33,45]. According to [37], they make up around two-thirds of all uni-relational data dependencies (those defined over a single relation schema) in practice. FDs form one of the most prolific database concepts. They are essential for the fundamental tasks of database modeling and design [25,106,108,117,118], normalization on the conceptual [32,90,107] and logical level [32,81,127], and maintenance [57,78,114]. They have proven applications in data cleaning [55,56], entry [106], exchange [99], integration [24,102,122], sampling [15,36], warehousing [83], indexing [77], information retrieval [76], uncertain data management [79,126], repairing [13] and consistent query answering [103], query optimization [26,42,104], access control [18], and privacy-preserving data publishing [123] among others. A basic prerequisite for addressing these fundamental tasks and taking effective advantage of these applications is that data engineers are able to identify the FDs that are meaningful for their application domain.

Example 1 Suppose that in designing an information system for a company, the team of data engineers has identified the relationship between employees and managers in departments as an important information unit. Therefore, the team has decided to utilize the relation schema EMPLOYMENT with attributes *Emp*, *Dept*, and *Mgr*. The schema stores the name of an employee in the *Emp* column, the name of the department in which the employee works in the *Dept* column, and the name of the department's manager in the *Mgr* column. The team has identified additional constraints they consider important for the schema EMPLOYMENT. Firstly, the information stored in the columns *Emp* and *Mgr* must be total, i.e., no occurrences of null values are permitted in these columns. Consequently, an SQL definition of the table may look as follows:

```
CREATE TABLE EMPLOYMENT (
    Emp VARCHAR NOT NULL,
    Dept VARCHAR,
    Mgr VARCHAR NOT NULL);
```

Moreover, the team of data engineers believes that every employee can work for only one department, and that every department has only one manager. These constraints can be expressed as the FDs $Emp \rightarrow Dept$ and $Dept \rightarrow Mgr$, respectively. The team would like to consult the experts of the application domain to find out whether their current design choice captures all the requirements necessary. In order to validate their own understanding of the application domain and to facilitate the knowledge acquisition from the domain experts, the team plans to create test data that represents concisely their current understanding of the database.

Example 1 illustrates the findings of previous research [2,9,15,35,36,96]: The creation of good test data can help data engineers with the challenge to consolidate the set of meaningful FDs. In fact, the findings suggest to use good test data to identify actually meaningful FDs that are incorrectly perceived as meaningless [9,15,36,96]. It is thus an important task to create good test data effectively and efficiently. Informally, good means here that the test data satisfies the FDs that the team of data engineers considers meaningful and violates the FDs the team perceives as meaningless.

More formally, given a set $\Sigma \cup \{\varphi\}$ of FDs, we say that Σ *implies* φ if every relation that satisfies all FDs in Σ also satisfies φ. We use Σ^* to denote the *semantic closure* of Σ, i.e., the set of all FDs implied by Σ. *Armstrong relations* have been identified as an effective means to represent concisely abstract sets of FDs in the form of a single relation. Indeed, Armstrong relations are widely regarded as good test data and a helpful tool for data engineers to judge, justify, convey, or test their understanding of the relation schema [2,9,15,35,36,96]. More precisely, an Armstrong relation for a set Σ of FDs is a single relation that satisfies every FD in Σ^* and violates every FD not in Σ^* [49]. Hence, an FD is implicit in the explicit specification of an FD set Σ if and only if it is satisfied by an Armstrong relation for Σ. For this reason, Armstrong relations represent one of the few instances where *example-based reasoning* is effective. That is, if Σ constitutes the design choice of the set of FDs currently perceived as meaningful to the underlying application domain, then an Armstrong relation r_Σ for Σ constitutes an example on which data engineers can test the meaningfulness of an arbitrary FD φ. Namely, φ is meaningful under the current design choice if and only if the example relation r_Σ satisfies φ. Consequently, this approach to design is called *design-by-example*. Empirical evidence has been presented that Armstrong relations help data engineers to identify meaningful FDs they perceived incorrectly as meaningless before inspecting an Armstrong relation [82]. Industry-leading data modeling tools, such as the *ERwin* data modeler, emphasize the need for good test data to validate the semantics of the models produced [23]. We exemplify these observations by the following case, expanding on Example 1.

Example 2 Consider the relation schema EMPLOYMENT with attribute set *Emp*, *Dept*, and *Mgr*, and FD set Σ that consists of $Emp \rightarrow Dept$ and $Dept \rightarrow Mgr$. As in Example 1, assume that only *Emp* and *Mgr* have been declared NOT NULL. In this case, the relation

Emp	*Dept*	*Mgr*
Hilbert	Math	Gauss
Pythagoras	Math	Gauss
Einstein	Physics	Gauss
Turing	ni	von Neumann
Turing	ni	Gödel

is an Armstrong relation for Σ and the NOT NULL constraints. Note that, according to the semantics under the *no information* interpretation of the null value ni [6,69,89], the relation satisfies the FD $Dept \rightarrow Mgr$ since the FD can only be violated if there are distinct tuples that are total and agree in the *Dept* column, and differ in the *Mgr* column. When the domain experts inspect this Armstrong relation, they are alerted to the fact that *Turing* has two managers *von Neumann* and *Gödel*. More specifically, the domain experts communicate to the team of data engineers that every employee should only occur once in the *Emp* column. Consequently, the inspection of the Armstrong relation has revealed that under the current design choice neither the meaningful FD $Emp \rightarrow Mgr$ nor the meaningful key *Emp* have been successfully captured. As a consequence, the team has now a choice of either specifying the FD $Emp \rightarrow Mgr$ additionally to the FDs in Σ, or declaring *Dept* as NOT NULL.

Example 2 illustrates the assistance that Armstrong relations can provide to data engineers in their task to capture the semantics of the underlying application domain. Unfortunately, the current toolbox of Armstrong relations does not apply to SQL tables where arbitrary attributes can be declared `NOT NULL`. That means that currently, there is no effective support in identifying the set of meaningful FDs over SQL tables. While FDs have been studied in this context [6,89], the concept of Armstrong relations has not been investigated yet. To address this research gap, we adopt the formal framework of Lien [89], Atzeni, and Morfuni [6] who have studied FDs under Zaniolo's *no information* interpretation of nulls. It is intuitive and well known that FDs interact quite differently in the presence of `NOT NULL` constraints than they do over total relations [6]. Therefore, it should come as no surprise that Armstrong relations for FD sets in the presence of `NOT NULL` constraints can be quite different from Armstrong relations for FDs over total relations. The following example illustrates that currently existing techniques for creating total Armstrong relations do not apply to real SQL table definitions.

Example 3 Consider the relation schema EMPLOYMENT with attributes *Emp*, *Dept* and *Mgr*, and FD set Σ that consists of $Emp \rightarrow Dept$ and $Dept \rightarrow Mgr$. Suppose we would apply the existing techniques to create the total Armstrong relation

Emp	*Dept*	*Mgr*
Hilbert	Math	Gauss
Pythagoras	Math	Gauss
Einstein	Physics	Gauss
Turing	Computer Science	von Neumann

for Σ [9]. Then, this is only an Armstrong relation for Σ in the special case where all attributes are `NOT NULL`, but not an Armstrong relation for the SQL table definition of Example 1. If we incorrectly used this total relation as a concise representation for our SQL table definition of Example 1, then, it would give us the false impression that the current design choice successfully captures both the meaningful key *Emp* and the meaningful FD $Emp \rightarrow Mgr$. Consequently, there is a real need to study Armstrong relations for FDs in the presence of arbitrary `NOT NULL` constraints.

Contributions and organization. In this article, we investigate the concept of Armstrong tables for FDs in the presence of `NOT NULL` constraints. Following previous research [6,69,89], we adopt Zaniolo, Lien, Atzeni, and Morfuni's *no information* interpretation of null values. This is the most primitive interpretation which allows data engineers to model non-existent information as well as existent information that is currently unknown. In this context, Atzeni and Morfuni have studied FDs in the presence of a null-free subschema (NFS). Essentially, an NFS is the subset of the underlying relation schema whose attributes have been declared `NOT NULL`. More specifically, Atzeni and Morfuni have presented an axiomatization for the implication of FDs in the presence of an NFS, and an algorithm that decides the corresponding implication problem in time linear in the input size [6]. Note that the combination of FDs and an NFS subsumes the uniqueness and (primary) key constraints over SQL relations.

The objective of this article is to extend the existing theory of Armstrong relations to a toolbox that is applicable to SQL table definitions with functional dependencies. We review previous work in Sect. 2. Subsequently, we define the underlying concepts and present related previous results in Sect. 3.

As a first contribution, we show in Sect. 4 that FDs do not enjoy Armstrong relations when null values are permitted to occur. That is, we identify relation schemata and sets of FDs for which no Armstrong relations exist. This is in contrast to the case of total relations. Fortunately, the source of this negative result are so-called non-standard FDs. These are FDs of the form $\emptyset \rightarrow A$ with an empty attribute set on the LHS. One may argue that these FDs do not occur in practice since they would enforce all entries in the A-column to be the same. We will show that the class of standard FDs does enjoy Armstrong relations, even in the presence of an arbitrary NFS. The situation is reminiscent of the situation for functional and inclusion dependencies over total relations where only standard FDs and inclusion dependencies enjoy Armstrong databases [52]. For these reasons, we will focus on the class of standard FDs in the presence of an NFS.

As a second contribution, we characterize Armstrong tables in Sect. 5. That is, we give sufficient and necessary conditions for a given relation to be an Armstrong table for a given set of standard FDs and a given NFS. Our characterization is based on extensions of the notions of *agree sets* [9], *maximal sets* [96] and "closed" sets from the special case of total relations. However, we demonstrate that in the presence of null values, the closure operation is not idempotent. Therefore, the maximal sets are no longer the intersection generators of closed sets, a property which was fundamental in the development of the results for the special case of total relations [9]. This observation constitutes a significant challenge on the development of our results.

As a third contribution, we establish in Sect. 6 an algorithm that given a relation schema R, an NFS R_s and a set Σ of standard FDs over R, computes an Armstrong table for Σ and R_s. The algorithm is a non-trivial extension of the algorithm by Mannila and Räihä for FDs over total relations [96]. It is based on the computation of attribute subsets that are maximal for a given attribute, i.e., maximal with the property that the attribute is not functionally dependent on the subset. The computation of the families of maximal attribute subsets is incremental in the given set of standard FDs. The families evolve conceptually differently than they do in the special case of total relations.

As a fourth contribution, we investigate the complexity of certain problems associated with Armstrong tables in Sect. 7. While the problem of finding an Armstrong table remains precisely exponential in general, the size of the Armstrong tables that our algorithm produces is shown to be at most quadratic in the size of a minimum-sized Armstrong table. We show that the size of Armstrong tables for a given set of FDs can be exponentially smaller than an optimal cover of Σ. Therefore, just for the reason of the size of a representation, data engineers should always consider both representations of business rules: as abstract FD sets and as Armstrong tables for these. Finally, we also show that the problem of deciding whether a given attribute set is a Codd key (i.e., it enforces totality and uniqueness) with respect to a given set of FDs and an NFS is NP-complete. Therefore, our results extend the existing toolbox of Armstrong relations to SQL table definitions with no loss in time efficiency and almost no loss in space efficiency over total relations.

As a fifth contribution, we demonstrate in Sect. 8 how our results carry over to i) schemata on which no Codd key has been specified and ii) Armstrong tables for standard FDs and an NFS in the world of all FDs. Concerning i) we need to study the combined class of uniqueness constraints and FDs in the presence of an NFS. Concerning ii) Armstrong tables for the world of all FDs must additionally violate all non-trivial non-standard FDs, in contrast to the world of standard FDs.

As a final contribution, we illustrate the impact of our results on database practice in Sect. 9. We show how Armstrong tables can assist data engineers in recognizing meaningful constraints that were previously incorrectly perceived as meaningless. The recognition of these constraints can result in advanced designs that guarantee the absence of data redundancy and update anomalies, ensure new opportunities for the efficient processing of queries, and allow security officers to gain an advanced understanding about possible inference attacks.

In summary, our contributions extend well-known results from total relations to SQL tables. Hence, the resulting toolbox can be applied to instances that occur in real database systems. Moreover, almost all of the nice properties previously established for the special case of total relations [5,6,9,38,89,96] are retained in the more general case. We conclude in Sect. 10 where we also discuss multiple directions of future work.

Proofs. Due to space restrictions, we have moved some of the proofs to the electronic appendix.

2 Related work

Data dependencies have been studied thoroughly in various data models, and we refer the reader to more detailed surveys [1,53,94,116].

2.1 Total relations

We begin with a summary of the work over total relations. Keys and FDs are concepts almost as old as the relational model of data itself [28,29]. A total relation over a relation schema R satisfies a key X if and only if it satisfies the functional dependency $X \to R$. Therefore, keys are subsumed by FDs. Armstrong established the first axiomatization of FDs over total relations [5], now known as the *Armstrong axioms*. In fact, Armstrong showed that the Armstrong axioms are even *strongly complete* for the implication of FDs, i.e., for an arbitrary relation schema and an arbitrary set of FDs on that schema, he constructed a single finite set of tuples which satisfies precisely all implied FDs. That is, the reason why such specific relations became known as *Armstrong relations*. In general, axiomatizations can be applied by data engineers to enumerate all implied data dependencies. In practice, such an enumeration is often desirable, e.g., to validate the correct specification of explicit knowledge, to design and fine-tune databases or to optimize queries. In particular, the completeness of the inference rules ensures that all opportunities of utilizing implicit knowledge for these purposes have been exploited. Furthermore, an analysis of the completeness argument can provide invaluable hints for finding algorithms that efficiently decide the associated implication problem, i.e., to decide for an arbitrarily given FD set $\Sigma \cup \{\varphi\}$ whether Σ implies φ. For FDs over total relations, implication can be decided in time linear in the total number of attributes that occur in the input [7,44]. Such decision algorithms complement the enumeration algorithm by a further reasoning capability that can make efficient, but only partial decisions about implicit knowledge. These decisions are only partial in the sense that the input to this algorithm must also contain a candidate for an implied functional dependency. In contrast, the enumeration algorithm simply lists all implied dependencies. The reason for the prominence of the implication problem is manifold. An algorithm for testing the implication of dependencies enables us to test whether two given sets of dependencies are equivalent or whether a given set of dependencies is redundant. A solution to these problems is a big step toward automated database schema design [10,11] which some researchers see as the ultimate goal in dependency theory [8]. Moreover, such an algorithm can be used in relational normalization theory and practice involving many normal form proposals [7,8,11,17,29], requirements engineering and schema validation [96], data mining [98], in database security [16,18], view maintenance [78] and in query optimization [26,42,43]. More recently, the implication problem has received a lot of attention in other data models as well [3,4,6,21,22,59,62,66–68,74,80,85,112,115,121,124,125,127]. New application areas involve data cleaning [54], data transformations

[34], consistent query answering [27], and data exchange [51,87,101] and data integration [24,102].

Armstrong relations constitute an invaluable tool for the validation of semantic knowledge, and a user-friendly representation of integrity constraints. Armstrong relations have been deeply studied for keys [38,75] and FDs [5,9,41,96]. In particular, the existence of Armstrong relations for FDs was shown by Armstrong [5], and Fagin [50] has shown the existence of Armstrong relations for a large class of data dependencies; however, classes of data dependencies do not necessarily enjoy Armstrong relations by any means. The structure of Armstrong relations for the class of FDs over total relations has mainly been investigated by Beeri, Fagin, Statman and Howard [9] and Mannila and Räihä [96]. In the current article, we will extend several of their results from the special case of total to partial relations. The properties of Armstrong relations have also been analyzed for various other classes of data dependencies [36,40,49,52,61,96]. An excellent survey on Armstrong databases is [49]. Over the last decades, the concept of Armstrong relations has been found useful in many database applications. For example, informative Armstrong databases are used for the *semantic sampling of existing databases* [15,36]. They constitute small subsets of an existing database that satisfy the same data dependencies. In particular, De Marchi and Petit [36] applied the concept of informative Armstrong relations to generate a sample that only used 0.6% of the tuples in an existing real-world database and satisfied the exact same set of data dependencies. The small size of this sample enabled data engineers to identify the FDs meaningful for the application domain. *Design aids* [35,96, 110] have been developed that utilize Armstrong databases to help judge, justify, convey or test the data engineers' understanding of a given relation schema. Recently, empirical evidence has been established that Armstrong relations help data engineers to recognize meaningful FDs that they were not able to recognize without the help of Armstrong relations [82]. Failure to identify some meaningful functional dependency also means that the output of a *requirements analysis* is afflicted with errors. Empirical studies show that more than half the errors which occur during systems development are requirements errors [46,100]. Requirements errors are also the most common cause of failure in systems development projects [46,113]. The cost of errors increases exponentially over the development life cycle: it is more than 100 times more costly to correct a defect post-implementation than it is to correct it during requirements analysis [20]. This suggests that it would be more effective to concentrate quality assurance efforts in the requirements analysis stage, in order to catch requirements errors as soon as they occur or to prevent them from occurring altogether [129]. Hence, there are also strong economic incentives to utilize Armstrong relations for the acquisition of meaningful FDs. Finally, Kolaitis et al. have established first correspondences between unique characterizations of *schema mappings* and the existence of Armstrong bases [2]. An Armstrong basis refers to a finite set of pairs consisting of a source instance and a target instance that is a universal solution for the source instance. Our results may open the way to establish further correspondences.

2.2 Partial relations

We will now comment on some of the work concerning data dependencies in the presence of nulls. One of the most important extensions of Codd's basic relational model [28] is incomplete information. This is mainly due to the high demand for the correct handling of such information in real-world applications. Approaches to deal with incomplete information comprise incomplete relations [31,72,86], or-relations [73,88,119] or fuzzy relations [112]. Here, we focus on incomplete relations.

In the literature, many kinds of null values have been proposed; for example, "missing" or "value unknown at present" [30,64,65], "non-existence" [95], "inapplicable" [65], "no information" [128] and "open" [63]. The most primitive is the "no information" interpretation that can be used to model every kind of missing or incomplete information, and its semantics is certainly simple and well understood [6]. Lien [89] investigated FDs in partial relations under this interpretation and established an axiomatization for this class. Interestingly, the transitivity rule, which is part of the Armstrong axioms, is no longer sound in this more general context. Atzeni and Morfuni established axiomatizations and linear-time algorithms for deciding the implication of the combined class of FDs and various existence constraints, including NFSs [6]. It is precisely this line of work we continue in this paper. Recently, Hartmann and Link [69] established an axiomatization and an almost linear-time algorithm for deciding the combined class of functional and multivalued dependencies and NFSs and showed the equivalence of the implication problem to that of a propositional fragment of Cadoli and Schaerf's S-3 logics [109]. Levene and Loizou introduced and axiomatized the classes of weak and strong FDs with respect to a possible world semantics [85]. The axiomatization of strong FDs is given by the Armstrong axioms, while weak FDs have the same axiomatization as the FDs of Lien [89], Atzeni and Morfuni [6]. However, weak FDs are different from FDs under the no information interpretation. Consider, for example, the schema EMPLOYMENT with attributes *Emp*, *Dept* and *Mgr*, and the relation r with the two tuples (Turing, Comp, von Neumann) and (Turing, `ni`, Gödel). The relation satisfies the weak functional dependency $Emp \rightarrow Dept$ since the null value `ni` can be replaced by Comp (there is a possible world in which $Emp \rightarrow Dept$ is satisfied). However, under the no information interpretation, the functional dependency $Emp \rightarrow Dept$ is violated by r. That is, the two tuples have some information on the attribute *Emp* and the information is the same, but the first tuple has some information for *Dept* while the second tuple has no information for *Dept*. Levene and Loizou also showed that the combined class of weak and strong FDs enjoys Armstrong relations [85]. Paulley axiomatized the implication of a class of strong and lax FDs which advance the query processing in SQL [104]. In particular, Paulley's lax FDs are different from "no information" FDs, e.g., the relation r above satisfies the lax, but not the "no information" FD $Emp \to Dept$. Another difference is that Paulley accounts for the possible existence of duplicate rows in an SQL table by including tuple identifiers.

Over the last decade, FDs have also received a lot of attention from the XML community [4,68,80,121,127]. However, the study of Armstrong data trees for XML FDs has only been preliminary so far. In fact, Hartmann et al. [68] characterize the existence of such Armstrong data trees for a particular class of XML FDs under certain assumptions on the underlying XML schema. The results of our present article may provide valuable information for the study of Armstrong data trees for many classes of XML FDs.

3 Preliminaries

In this section, we summarize the basic notions required for our treatment of FDs over SQL tables. Our development will follow the extension of Codd's relational model of data [28] to encompass incomplete information by the *no information* null value introduced by Zaniolo [128], applied to FDs by Lien [89], to FDs and null existence constraints by Atzeni and Morfuni [6], and to functional and multivalued dependencies and a null-free subschema by Hartmann and Link [69].

3.1 Total and partial relations

Let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called attributes. A *relation schema* is a non-empty, finite subset R of $\mathfrak{A}$ whose attributes represent column names of a relation. Each attribute A of a relation schema R is associated with an infinite domain $dom(A)$, which represents the set of possible values that can occur in the column named A. In order to encompass incomplete information, it is assumed that the domain of every attribute has a null value distinct from all the other domain values, denoted by $\mathtt{ni} \in dom(A)$. In the literature, many kinds of null values have been proposed; for example, "missing" or "value unknown at present" [30,64,65], "non-existence" [95], "inapplicable" [65], "no information" [128] and "open" [63]. The intention of the null value $\mathtt{ni}$ is to mean "no information". That is, the null value $\mathtt{ni}$ associated with an attribute in a tuple means that no information is available about that attribute for that tuple. This is the most primitive interpretation but can be used to model every kind of missing or incomplete information, and its semantics is certainly simple and well understood [6].

If X and Y are sets of attributes, then we may write XY for $X \cup Y$. If $X = \{A_1, \ldots, A_m\}$, then we may write $A_1 \cdots A_m$ for X. In particular, we may write simply A to represent the singleton $\{A\}$. A *tuple* over R (R-tuple or simply tuple, if R is understood) is a function $t : R \to \bigcup_{A \in R} dom(A)$ with $t(A) \in dom(A)$ for all $A \in R$. For $X \subseteq R$, let $t[X]$ denote the restriction of the tuple t over R to X, and let $dom(X) = \prod_{A \in X} dom(A)$ denote the Cartesian product of the domains of attributes in X. A (partial) *relation* r over R is a finite set of tuples over R. Let t_1 and t_2 be two tuples over R. It is said that t_1 *subsumes* t_2 if for every attribute $A \in R, t_1[A] = t_2[A]$ or $t_2[A] = \mathtt{ni}$ holds. Intuitively, the information content of t_1 subsumes that one of t_2. In keeping consistency with previous work [6,89,91,128], the following restriction will be imposed: No relation in the database shall contain two tuples t_1 and t_2 such that t_1 subsumes t_2. Without null values, this amounts to saying that no two tuples are identical, an explicit requirement for database relations. The results we develop in this article for Armstrong tables and the class of FDs, however, remain valid even when we lift this restriction. For a more detailed analysis, see Sect. 8.

For a set $X \subseteq R$, a tuple t over R is X-total, if for all $A \in X, t[A] \neq \mathtt{ni}$. A relation r over R is X-total, if every tuple t of r is X-total. A relation r over R is a *total relation*, if it is R-total.

3.2 Agree sets of tuples

Over total relations, Mannila, Räihä, Beeri, Dowd, Fagin and Statman used the notion of an agree set of tuples for the investigation of Armstrong relations. Over partial relations, this notion requires a refinement that we introduce in the next definition.

Definition 1 Let R be some relation schema. For two tuples t, t' over R, let

$$\begin{aligned} ag(t,t') := \{(X,Y) \mid \forall A \in R(& \\ &((t[A] = t'[A] \wedge t[A] \neq \mathtt{ni}) \leftrightarrow A \in X) \wedge \\ &(t[A] = t'[A] \leftrightarrow A \in Y))\}\ . \end{aligned}$$

denote the *agree set* of t and t'. For a relation r over R, let $ag(r) = \{ag(t,t') \mid t, t' \in r \wedge t \neq t'\}$ denote the *agree set* of r, let $ag^s(r) = \{X \mid (X,Y) \in ag(r)\}$ denote the *strong agree set* of r, and let $ag^w(r) = \{Y \mid (X,Y) \in ag(r)\}$ denote the *weak agree set* of r. Finally, for $X \in ag^s(r)$ let $w(X) = \bigcap\{Y \mid (X,Y) \in ag(r)\}$.

Remark 1 Over total relations [9,96], it suffices to define $ag(t,t') := \{A \in R \mid t[A] = t'[A]\}$ since a total relation r satisfies $ag^s(r) = ag^w(r)$, and $w(X) = X$ for all

$X \in ag^s(r)$. Note that all elements of $ag(r)$ have the form (X, X) in the case of total relations.

Example 4 Let r denote the relation from Example 2:

Emp	*Dept*	*Mgr*
Hilbert	Math	Gauss
Pythagoras	Math	Gauss
Einstein	Physics	Gauss
Turing	ni	von Neumann
Turing	ni	Gödel

The agree set of the last two tuples is

$(\{Emp\}, \{Emp, Dept\})$

and the remaining agree sets of r are the following:

- $(\{Dept, Mgr\},\{Dept, Mgr\})$
- $(\{Mgr\},\{Mgr\})$
- $(\emptyset, \emptyset)$.

Specifically, $w(Emp) = \{Emp, Dept\}$.

3.3 FDs and null-free subschemata

FDs between sets of attributes have always played a central role in the study of relational databases [7,11,28] and seem to be central for the study of database design in other data models as well [4,66,80,85,115,118,121,124,125]. In relational databases, the notion of a functional dependency is well understood and the semantic interaction between these dependencies has been syntactically captured by Armstrong's axioms [5]. Lien defined the semantics of a *functional dependency* over partial relations [89]. Here, we state this definition in terms of our agree sets.

Definition 2 A *functional dependency* (FD) over the relation schema R is a statement $X \to Y$ where $X, Y \subseteq R$. The FD $X \to Y$ over R is satisfied by a relation r over R, denoted by $\models_r X \to Y$, if and only if for all $t_1, t_2 \in r$ the following holds: if $X \subseteq ag^s(t_1, t_2)$, then $Y \subseteq ag^w(t_1, t_2)$. FDs of the form $\emptyset \to Y$ are called *non-standard*, otherwise they are called *standard*.

Hence, whenever two tuples agree on a non-null restriction to X, then they also agree on the restriction to Y, which may be partial. In other words, an FD $X \to Y$ can only be violated by a relation r if there are two distinct X-total tuples $t_1, t_2 \in r$ such that $t_1[X] = t_2[X]$ and there is some attribute $A \in Y$ such that $t_1[A] \neq t_2[A]$ (i.e., either $t_1[A]$ and $t_2[A]$ have different non-null values, or one of $t_1[A]$ and $t_2[A]$ is null but the other value is non-null).

For total relations, the definition of an FD reduces to that of a functional dependency, and so is a correct generalization of the concept. It is also consistent with the no information interpretation [6]. Firstly, tuples with nulls in attributes in X cannot cause a violation of a functional dependency $X \to Y$: the nulls mean that no information is available about those attributes. Secondly, the functional dependence of Y on X forces any two X-total tuples t_1, t_2 where $t_1[X] = t_2[X]$ to have the same information on all the attributes in Y. That is, for all $A \in Y$, we have either $t_1[A] = \texttt{ni} = t_2[A]$ or $\texttt{ni} \neq t_1[A] = t_2[A] \neq \texttt{ni}$. One could also define the semantics differently, e.g., by saying that (i) for all $t_1, t_2 \in r$: if t_1, t_2 are both XY-total and $X \subseteq ag^s(t_1, t_2)$, then $Y \subseteq ag^s(t_1, t_2)$, or (ii) for all $t_1, t_2 \in r$: if $X \subseteq ag^s(t_1, t_2)$, then $Y \subseteq ag^s(t_1, t_2)$. However, option (i) removes the functional dependence of Y on X by stipulating that any tuples that are not Y-total cannot contribute to a violation of the FD $X \to Y$, independently of their values on X. Moreover, option (ii) makes the unnecessarily strict requirement that any tuples t_1, t_2 with $X \subseteq ag^s(t_1, t_2)$ must also be Y-total.

According to Atzeni and Morfuni [6], a *null-free subschema* (NFS) over the relation schema R is a an expression R_s where $R_s \subseteq R$. The NFS R_s over R is satisfied by a partial relation r over R, denoted by $\models_r R_s$, if and only if for all $t \in r$ and for all $A \in R_s$ we have $t[A] \neq \texttt{ni}$. Hence, the satisfaction of a null-free subschema R_s over R requires that partial relations over R are R_s-total. Without loss of generality, we can assume that on each relation schema a single NFS is defined. Null-free subschemata occur naturally in database practice: SQL allows one to specify attributes as `NOT NULL`, cf. Example 1. Consequently, the set of such attributes over a table definition would form the single null-free subschema over this table.

For a set Σ of constraints over some relation schema R, we say that a (partial) relation r over R *satisfies* Σ, denoted by $\models_r \Sigma$, if r satisfies every element of Σ. If for some $\sigma \in \Sigma$, the relation r does not satisfy σ we sometimes say that r violates σ (in which case r also violates Σ) and write $\not\models_r \sigma$ ($\not\models_r \Sigma$).

Example 5 For Example 1, we obtain the relation schema

$\text{EMPLOYMENT} = \{Emp, Dept, Mgr\}$

with NFS $\text{EMPLOYMENT}_s = \{Emp, Mgr\}$. The relation

Emp	*Dept*	*Mgr*
Turing	ni	von Neumann
Turing	ni	Gödel

satisfies EMPLOYMENT_s as well as the FDs $Emp \to Dept$ and $Dept \to Mgr$. However, the relation violates the FD $Emp \to Mgr$.

3.4 Implication and inference

For the design, maintenance and applications of a relational database, data dependencies are identified as semantic constraints on the relations which are intended to be instances of the database schema. During the design process or lifetime of a database, one usually needs to determine further dependencies which are logically implied by the given ones. In line with the literature of database constraints, we restrict our attention to the implication of constraints in some fixed class $\mathcal{C}$: FDs in the presence of an NFS.

Let R be a relation schema, let $R_s \subseteq R$ denote an NFS over R, and let $\Sigma \cup \{\varphi\}$ be a set of FDs over R. We say that Σ*implies*φ in the presence of R_s, denoted by $\Sigma \models_{R_s} \varphi$, if every relation r over R that satisfies Σ and R_s also satisfies φ. If Σ does not imply φ in the presence of R_s we may also write $\Sigma \not\models_{R_s} \varphi$.

The *implication problem for functional dependencies in the presence of a null-free subschema* is to decide, given any relation schema R, any NFS R_s over R, and any set $\Sigma \cup \{\varphi\}$ of FDs over R, whether $\Sigma \models_{R_s} \varphi$. For the class of FDs in the presence of an NFS, the sets $\Sigma \cup \{\varphi\}$ over a relation schema R are always finite. Moreover, if $R_s = \emptyset$ we also write $\Sigma \models \varphi$ instead of $\Sigma \models_{\emptyset} \varphi$. This covers the case where every attribute is NULL [89]. The case where every attribute is NOT NULL is covered when $R_s = R$.

Note that for FDs (in the presence of an NFS), it does not matter whether we restrict our attention to relations that are finite, i.e., the implication problem coincides with the finite implication problem where only finite relations are considered [1]. For this reason, we will only speak of the implication problem.

For an FD set Σ over a relation schema R and an NFS R_s over R, let the FD set $\Sigma^*_{R_s} = \{\varphi \mid \Sigma \models_{R_s} \varphi\}$ denote the *semantic closure* of Σ. For a finite FD set $\Sigma \cup \{\varphi\}$ and a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote an *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of FDs such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or results from an application of an inference rule in $\mathfrak{R}$ to some FDs in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. For a finite FD set Σ let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ denote the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. $\mathfrak{R}$ is said to be *sound* (*complete*) for the implication of FDs in the presence of an NFS if for every relation schema R, for every NFS R_s over R and for every FD set Σ over R we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{R_s}$ ($\Sigma^*_{R_s} \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* for the implication of FDs in the presence of an NFS if $\mathfrak{R}$ is both sound and complete for the implication of FDs in the presence of an NFS.

Example 6 Consider the relation schema EMPLOYMENT with the NFS EMPLOYMENT$_s$ from Example 5 again. Let Σ denote the set of FDs over EMPLOYMENT that consists of *Emp* $\to$ *Dept* and *Dept* $\to$ *Mgr*. Since the relation in Example 5 satisfies the NFS EMPLOYMENT$_s$ as well as Σ, but it violates the FD *Emp* $\to$ *Mgr* we conclude that *Emp* $\to$ *Mgr* is not in the semantic closure $\Sigma^*_{\text{EMPLOYMENT}_s}$ of Σ.

Table 1 Axiomatization of functional dependencies in the presence of a null-free subschema R_s

$\dfrac{}{XY \to X}$ (reflexivity)	$\dfrac{X \to Y;\ X \to Z}{X \to YZ}$ (union)	$\dfrac{X \to YZ}{X \to Y}$ (decomposition)
	$\dfrac{X \to Y;\ Y \to Z}{X \to Z}\ Y - X \subseteq R_s$ (null transitivity)	

Remark 2 It was shown recently [69] that an FD set Σ implies an FD φ in the presence of the NFS R_s if and only if the set Σ' of Horn clauses resulting from Σ logically implies the Horn clause φ' resulting from φ in Cadoli and Schaerf's para-consistent family of S-3 logics. Here, S consists of those propositional variables that correspond to the attributes of the NFS R_s and must be interpreted as either *true* or *false*, whereas variables outside of S can also be interpreted as both *true* and *false*. Importantly, the truth tables that define the logical operators of negation, disjunction and conjunction are the tables that Codd suggested for extending the relational algebra by means of a three-valued logic and the null substitution principal, and which has been adopted by SQL [31].

Atzeni and Morfuni have established a finite axiomatization [6] for the implication of FDs in the presence of an NFS. The inference rules are given in Table 1.

Note that the null transitivity rule can only infer the FD $X \to Z$ from the FDs $X \to Y$ and $Y \to Z$, if all the attributes in $Y - X$ have been declared NOT NULL, i.e., are members of the NFS R_s. Also, note that the so-called *augmentation rule*

$$\frac{X \to Y}{XZ \to Y}$$

follows from the reflexivity axiom and the null transitivity rule [6].

Example 7 Consider the relation schema EMPLOYMENT with the NFS EMPLOYMENT$_s$ from Example 5 again. Then, the FD *Emp* $\to$ *Mgr* cannot be inferred from the two FDs *Emp* $\to$ *Dept* and *Dept* $\to$ *Mgr* by means of the *null transitivity* rule since the attribute *Dept* is not an element of EMPLOYMENT$_s$.

Atzeni and Morfuni also established a linear-time algorithm for deciding the implication problem for FDs in the presence of an NFS[6]. As Beeri and Bernstein did for total relations [7], Atzeni and Morfuni utilized the notion of an *attribute closure*

$$X^*_{\Sigma,R_s} = \{A \in R \mid \Sigma \models_{R_s} X \to A\}$$

of an attribute set X with respect to an FD set Σ and an NFS R_s over the relation schema R [6]. An FD $X \twoheadrightarrow Y$ over R is implied by Σ in the presence of R_s if and only if $Y \subseteq X^*_{\Sigma,R_s}$ holds [6]. Algorithm 1 computes the attribute closure X^*_{Σ,R_s} of X with respect to Σ and R_s over R [6].

Algorithm 1 (NFSClosure(X, Σ, R_s, R))

Input: attribute subset X, FD set Σ, NFS R_s all over R, relation schema R
Output: attribute closure X^*_{Σ,R_s} of X with respect to Σ and R_s
Method:

```
(A0) CLOSURE := $X$;
(A1) repeat
        OLDCLOSURE := CLOSURE;
        for all $V \to W \in \Sigma$ do
          if $V \subseteq$ CLOSURE $\cap XR_s$ then
            CLOSURE := CLOSURE $\cup W$;
          endif;
        enddo;
      until OLDCLOSURE = CLOSURE;
(A2) return CLOSURE;
```

Algorithm 1 corrects the algorithm originally proposed in [6]. For example, consider the relation schema EMPLOYMENT with the NFS $\{Dept\}$ and let Σ consist of the two FDs $Emp \to Dept$ and $Dept \to Mgr$. On input

$(\{Emp\}, \Sigma, \{Dept\}, \text{EMPLOYMENT})$

the original algorithm [6] returns the set $\{Emp,Dept\}$, but the correct result is the set $\{Emp,Dept,Mgr\}$.

Example 8 Consider the relation schema EMPLOYMENT with the NFS EMPLOYMENT$_s$ from Example 5 again, and let Σ consist of the two FDs $Emp \to Dept$ and $Dept \to Mgr$. On input

$(\{Emp\}, \Sigma, \text{EMPLOYMENT}_s, \text{EMPLOYMENT})$

Algorithm 1 returns the set $\{Emp,Dept\}$. Consequently, the FD $Emp \to Mgr$ is not implied by Σ in the presence of the NFS EMPLOYMENT$_s$ since Mgr is not an element of $\{Emp\}^*_{\Sigma,\text{EMPLOYMENT}_s} = \{Emp, Dept\}$.

3.5 Uniqueness constraints and Codd keys

Functional dependencies in the presence of arbitrary null-free subschemata subsume SQL's uniqueness constraint as well as so-called Codd keys.

A *uniqueness constraint* over a relation schema R is an expression $unique(X)$ with $X \subseteq R$. A relation r over R satisfies $unique(X)$ if and only if for all distinct $t, t' \in r$ the following holds: if t and t' are X-total, then $t[X] \neq t'[X]$. It follows that r satisfies $unique(X)$ if and only if r satisfies the FD $X \to R$.

Moreover, a *Codd key* over a relation schema R is an expression $Codd(X)$ with $X \subseteq R$. A relation r over R satisfies $Codd(X)$ if and only if r is X-total and for all distinct $t, t' \in r$ it is true that $t[X] \neq t'[X]$. From this definition, it follows that r satisfies $Codd(X)$ if and only if r satisfies the NFS X and satisfies the FD $X \to R$.

It is now relatively easy to see that we can express uniqueness constraints and Codd keys by means of FDs and an NFS.

Theorem 2 *Let Σ be a set of FDs and let R_s be an NFS over R. Then the following hold*:

1. $\Sigma \models_{R_s} unique(X)$ *if and only if* $\Sigma \models_{R_s} X \to R$,
2. $\Sigma \models_{R_s} Codd(X)$ *if and only if* $\Sigma \models_{R_s} X \to R$ *and* $X \subseteq R_s$.

3.6 Armstrong relations

Let Σ be a set of constraints in class $\mathcal{C}$ over some relation schema R. A relation r over R is said to be an *Armstrong relation* for Σ, if for all φ in $\mathcal{C}$ over R it is true that r satisfies φ if and only if Σ implies φ. Hence, r is a perfect representation of the constraint set Σ in the sense that it satisfies all the constraints implied by Σ, but violates all the constraints not implied by Σ.

A class $\mathcal{C}$ of constraints is said to *enjoy Armstrong relations* if and only if for every relation schema R, and for every set Σ of constraints in $\mathcal{C}$ over R, there is some relation r over R that is an Armstrong relation for Σ.

In this paper, we are interested in the class $\mathcal{C}$ of FDs in the presence of an NFS, which permits arbitrary FD sets together with a single arbitrary null-free subschema. In this context, we also speak of *Armstrong tables* instead of Armstrong relations. Note that an NFS R_s implies another NFS R'_s if and only if $R'_s \subseteq R_s$ holds.

Example 9 The relation of Example 3 is an Armstrong table for the FD set

$\Sigma = \{Emp \to Dept, Dept \to Mgr\}$

and the NFS $R_s = R = \{Emp, Dept, Mgr\}$. Moreover, the relation of Example 2 is an Armstrong table for the FD set Σ and the NFS $R_s = \{Emp, Mgr\}$ over $R = \{Emp, Dept, Mgr\}$. The relation of Example 2 is not an Armstrong table for Σ and $R_s = R$, and the relation of Example 3 is not an Armstrong table for Σ and $R_s = \{Emp, Mgr\}$: it satisfies $R'_s = R$ and the FD $Emp \to Mgr$, but neither is implied by Σ and R_s.

Further outline. The development of these notions have led us to several questions. A first fundamental question is whether the class of FDs in the presence of an NFS enjoys Armstrong tables. We will pursue a first answer to this question in Sect. 4. Another question is the following: how can we validate that the relations in Examples 2 and 3 are indeed Armstrong tables? More generally, we would like to have a simple characterization that allows us to judge whether any given relation is an Armstrong table for an arbitrarily given set of FDs and an arbitrarily given NFS R_s. Such characterizations will be established in Sect. 5. These will assist us in Sect. 6 when we develop an algorithm that computes an Armstrong table for an arbitrarily given FD set and an arbitrarily given NFS. We investigate questions regarding the complexity of Armstrong tables in Sect. 7.

4 FDs do not enjoy Armstrong tables

For total relations, i.e., in the special case where the null-free subschema contains all the attributes of the underlying relation schema, it is well known that FDs enjoy Armstrong relations [5,9,49]. We will show now, however, that the class of FDs does not enjoy Armstrong tables if there are attributes that are not declared `NOT NULL`. Intuitively in such a case, non-standard FDs force all the tuples of a relation to have the same value on some attribute. For relations to be Armstrong, however, it may also be required that the values of such an attribute do differ for some distinct tuples. Consequently, Armstrong tables cannot exist in general.

Theorem 3 *The class of functional dependencies in the presence of a null-free subschema does not enjoy Armstrong tables.*

Proof We show that there is a relation schema R, an NFS R_s and an FD set Σ over R such that there is no Armstrong table for Σ and R_s.

Let $R = ABC$, $R_s = BC$ and let $\Sigma = \{\emptyset \to A, A \to B\}$. First we show that an Armstrong table r for Σ and R_s must violate the FDs $AB \to C$ and $C \to B$. In fact, the relation

A	B	C
a	b	c
a	b	c'

satisfies the FD set Σ and the NFS R_s, but violates the FD $AB \to C$. Consequently, $\Sigma \not\models_{R_s} AB \to C$. Moreover, the relation

A	B	C
ni	b'	c
ni	b	c

satisfies the FD set Σ and the NFS R_s, but violates the FD $C \to B$. Consequently, $\Sigma \not\models_{R_s} C \to B$. We conclude that an Armstrong table for Σ in the presence of R_s must at least satisfy Σ and violate the FDs $AB \to C$ and $C \to B$.

We show now that any relation that satisfies the FD $\emptyset \to A$ and violates the FDs $AB \to C$ and $C \to B$ must violate the FD $A \to B$. Let r be a relation that satisfies the FD $\emptyset \to A$ and violates the FDs $AB \to C$ and $C \to B$. Since r violates $AB \to C$ there are two distinct tuples $t_1, t_2 \in r$ such that $t_1[AB] = t_2[AB]$ and t_1, t_2 are both AB-total, and $t_1[C] \neq t_2[C]$. Since r violates the FD $C \to B$ there are two distinct tuples $t_3, t_4 \in r$ such that $t_3[C] = t_4[C]$ and t_3, t_4 are both C-total, and $t_3[B] \neq t_4[B]$. Since r satisfies the FD $\emptyset \to A$, it follows that $t_3[A] = t_4[A] = t_1[A]$, and, in particular, that t_3 and t_4 are A-total. Hence, the tuples t_3, t_4 witness that r violates the FD $A \to B$. Consequently, r cannot be an Armstrong table for Σ and R_s.

For the proof of Theorem 3, it actually suffices to consider the FD $AB \to C$ and the null-free subschema $R_s = BC$. A relation that violates the FD $AB \to C$ must contain two tuples that are A-total. A relation that is Armstrong for Σ and R_s should contain a tuple that carries the no information null value `ni` on A. Consequently, no relation can violate the FD $AB \to C$ and the NFS A, and satisfy the FD $\emptyset \to A$. Hence, there is no Armstrong table for Σ and R_s.

However, the proof of Theorem 3 also shows that Lien's class of FDs over partial relations [89] does not enjoy Armstrong relations. That is, the proof argument also applies when no NFS R_s is present. In particular, note that $\Sigma \not\models AB \to C$ and $\Sigma \not\models C \to B$ also hold.

Corollary 1 *Lien's class of FDs does not enjoy Armstrong relations.*

The next two examples illustrate the difference between the special case of total relations (Example 10) and the general case (Example 11) we consider here.

Example 10 Consider the relation schema

$\text{EMPLOYMENT} = \{Emp, Dept, Mgr\}$,

the NFS $R_s = \{Emp, Dept, Mgr\}$ and let Σ consist of the two FDs $\emptyset \to Mgr$ and $Mgr \to Dept$. The relation r

Emp	*Dept*	*Mgr*
Hilbert	Maths	Gauss
Pythagoras	Maths	Gauss

is an Armstrong table for Σ and R_s. In particular, note that the FD $\emptyset \to Dept$ is implied by Σ and R_s.

Example 11 Consider the relation schema

$\text{EMPLOYMENT} = \{Emp, Dept, Mgr\}$,

the NFS $R_s = \{Emp, Dept\}$ and let Σ consist again of the two FDs $\emptyset \to Mgr$ and $Mgr \to Dept$. This is the case of the proof in Theorem 3. Here, the relation r'

Emp	*Dept*	*Mgr*
Gauss	Maths	ni
Gauss	Physics	ni
Turing	Computers	Gödel
von Neumann	Computers	Gödel

is not an Armstrong table for Σ and R_s. Indeed, it violates all FDs not implied by Σ and R_s and satisfies all FDs implied by Σ and R_s except $\emptyset \to Mgr$.

While the result of Theorem 3 is negative in general, we will see that Armstrong tables do exist for the class of standard FDs in the presence of an NFS. Since it can be argued that non-standard FDs rarely occur in practice, the situation is actually rather pleasant. It is reminiscent of Fagin and Vardi's result that the class of standard FDs and inclusion dependencies over total relations enjoys Armstrong databases, while the class of FDs and inclusion dependencies does not [52].

5 Characterization of Armstrong tables

In this section, we continue Lien [89], Atzeni and Morfuni's study [6] of the class of FDs in the context of partial relations and NFSs, respectively. As a first main result, we extend Mannila, Räihä, Beeri, Dowd, Fagin and Statman's characterization of Armstrong relations for FDs from total relations [9,96]. Our generalization requires an extension of Demetrovics' notion of maximal sets of attributes [38,96] and uses a refinement of Mannila, Räihä, Beeri, Dowd, Fagin and Statman's notion of an agree set of tuples, already given in Definition 1. Due to Theorem 3, we will focus in Sects. 5, 6 and 7 on the class $\mathcal{C}$ of standard FDs. In Sect. 8, we will look at extensions of our results.

5.1 Maximal families of sets

Definition 3 Let Σ be a set of standard FDs and let R_s be an NFS over a relation schema R. For an attribute $A \in R$ we define the *maximal set* $max_{\Sigma,R_s}(A)$ of A with respect to Σ and R_s as follows:

$$max_{\Sigma,R_s}(A) := \{\emptyset \neq X \subseteq R \mid \Sigma \not\models_{R_s} X \to A \wedge \\ \forall B \in R - X(\Sigma \models_{R_s} XB \to A)\}.$$

The *maximal sets* of R with respect to Σ and R_s are defined as $max_{\Sigma,R_s}(R) = \bigcup_{A\in R} max_{\Sigma,R_s}(A)$. If Σ and R_s are clear from the context, we may simply write $max(A)$ and $max(R)$, respectively.

Thus, the maximal sets of an attribute A with respect to Σ and R_s are the maximal attribute subsets of the underlying relation schema R that do not functionally determine A.

Example 12 Let $\Sigma = \{Emp \to Dept, Dept \to Mgr\}$ and let $R_s = \{Emp, Dept, Mgr\}$ be an NFS over the relation schema EMPLOYMENT. Then $max_{\Sigma,R_s}(Emp)$ consists of $\{Dept, Mgr\}$, $max_{\Sigma,R_s}(Dept)$ contains the single element $\{Mgr\}$, and $max_{\Sigma,R_s}(Mgr) = \emptyset$.

For $R_s = \{Emp, Mgr\}$, $max_{\Sigma,R_s}(Emp)$ contains the single element $\{Dept, Mgr\}$, $max_{\Sigma,R_s}(Dept)$ contains the single element $\{Mgr\}$, and $max_{\Sigma,R_s}(Mgr)$ contains the singleton $\{Emp\}$.

5.2 Characterizations

For a relation r over a relation schema R let

$$total(r) := \{A \in R \mid \forall t \in r(t[A] \neq \texttt{ni})\}$$

denote the set of those attributes A of R such that no tuple t of r carries a null value ni on A.

Theorem 4 *Let R be some relation schema, let Σ be a set of standard FDs and let R_s be an NFS over R. For all relations r over R it holds that r is an Armstrong table for Σ and R_s if and only if both of the following conditions are satisfied:*

1. *for all non-empty $X \subseteq R$ we have*

   $$X^*_{\Sigma,R_s} = \bigcap\{w(Z) \mid X \subseteq Z \in ag^s(r)\}, \text{ and}$$

2. $total(r) = R_s$.

For the special case of total relations, i.e., where $R_s = R$, it is well known that every FD set Σ and the NFS R over R defines a closure operator $(\cdot)^*_{\Sigma,R} : \mathcal{P}(R) \to \mathcal{P}(R)$ by mapping every attribute subset $X \subseteq R$ to its attribute closure $X^*_{\Sigma,R}$. In fact, $(\cdot)^*_{\Sigma,R_s}$ is a closure operator since it is *extensive* ($X \subseteq X^*_{\Sigma,R}$), *increasing* (for $X \subseteq Y$ we have $X^*_{\Sigma,R} \subseteq Y^*_{\Sigma,R}$) and *idempotent* $\left(\left(X^*_{\Sigma,R}\right)^*_{\Sigma,R} = X^*_{\Sigma,R}\right)$. In particular, an attribute subset X of R is said to be *closed* with respect to the FD set Σ and the NFS R if $X^*_{\Sigma,R} = X$. The set of all subsets of R that is closed with respect to the FD set Σ and the NFS R is denoted by $cl_{\Sigma,R}(R)$. For the special case where $R_s = R$, it is well known that $cl_{\Sigma,R_s}(R)$ is closed under intersection. Thus, there is a unique minimal subfamily of *generators* $gen_{\Sigma,R}(R) \subseteq cl_{\Sigma,R}(R)$ such that each member of $cl_{\Sigma,R}(R)$ can be expressed as an intersection of sets in $gen_{\Sigma,R}(R)$ [9].

Beeri et al. [9] have shown that r is an Armstrong table for an FD set Σ and the NFS R over R if and only if $gen_{\Sigma,R}(R) \subseteq ag^s(r) \subseteq cl_{\Sigma,R}(R)$ holds. Later, Mannila and Räihä [96] have shown that for an arbitrary relation schema R, an arbitrary FD set Σ and the special NFS R over R it is true that $max_{\Sigma,R}(R) = gen_{\Sigma,R}(R)$.

If we allow arbitrary null-free subschemata R_s of R, then the situation is different. In particular, $(\cdot)^*_{\Sigma,R_s} : \mathcal{P}(R) \to \mathcal{P}(R)$ is no longer idempotent, and therefore no closure operator, as the next example illustrates.

Example 13 Let R denote the relation schema EMPLOYMENT from Example 2 where the FD set Σ consists of the two FDs $Emp \to Dept$ and $Dept \to Mgr$, and the NFS R_s consists of the attributes *Emp* and *Mgr*. Then, the attribute subset closures are the following:

- $\emptyset^*_{\Sigma,R_s} = \emptyset$,
- $\{Emp\}^*_{\Sigma,R_s} = \{Emp, Dept\}$,
- $\{Dept\}^*_{\Sigma,R_s} = \{Dept, Mgr\}$,
- $\{Mgr\}^*_{\Sigma,R_s} = \{Mgr\}$,
- $\{Emp, Dept\}^*_{\Sigma,R_s} = R$,
- $\{Emp, Mgr\}^*_{\Sigma,R_s} = R$,
- $\{Dept, Mgr\}^*_{\Sigma,R_s} = \{Dept, Mgr\}$, and
- $\{Emp, Dept, Mgr\}^*_{\Sigma,R_s} = R$.

Hence, we can see that the attribute subset closure $(\cdot)^*_{\Sigma,R_s}$ is not idempotent, and

$cl_{\Sigma,R_s}(R) = \{\emptyset, \{Mgr\}, \{Dept, Mgr\}\} = gen_{\Sigma,R_s}(R)$,

is different from

$max_{\Sigma,R_s}(R) = \{\{Emp\}, \{Mgr\}, \{Dept, Mgr\}\}$.

Since cl_{Σ,R_s} does not define a closure operator for arbitrary NFS R_s over R the concept of *closed attribute subsets* is no longer useful in this context. For this reason, we will utilize the maximal set families $max_{\Sigma,R_s}(R)$ to characterize the situation when an arbitrary relation is an Armstrong table for a given FD set Σ and a given NFS R_s over the relation schema R.

Theorem 5 *Let R be some relation schema, let Σ be a set of standard FDs and let R_s be an NFS over R. For all relations r over R it holds that r is an Armstrong table for Σ and R_s if and only if the following conditions are satisfied:*

1. $\forall A \in R \forall X \in max_{\Sigma,R_s}(A)(X \in ag^s(r) \wedge A \notin w(X))$,
2. $\forall X \in ag^s(r)(X^*_{\Sigma,R_s} \subseteq w(X))$, *and*
3. $total(r) = R_s$.

Remark 3 Theorem 5 generalizes the characterization of Armstrong relations for standard FDs from the special case where the NFS R_s is R. In fact, if r is total, then condition 1. says that for all $A \in R$ and for all $X \in max_{\Sigma,R_s}(A)$ we have $X \in ag^s(r)$ (and that $A \notin X$ which follows from $X \in max_{\Sigma,R_s}(A)$). That is, condition 1. says that $max_{\Sigma,R_s}(R) \subseteq ag^s(r)$. Furthermore, condition 2. says that $\forall X \in ag^s(r)(X^*_{\Sigma} \subseteq X)$, i.e., all (strong) agree sets are closed: $ag^s(r) \subseteq cl_{\Sigma,R_s}(R)$.

The following example illustrates Theorem 5.

Example 14 Let EMPLOYMENT and the FD set

$\Sigma = \{Emp \rightarrow Dept, Dept \rightarrow Mgr\}$

as before, but let $R_s = \{Emp, Mgr\}$. The following relation r:

Emp	*Dept*	*Mgr*
Hilbert	Math	Gauss
Pythagoras	Math	Gauss
Einstein	Physics	Gauss
Turing	`ni`	von Neumann
Turing	`ni`	Gödel

is an Armstrong table for Σ and R_s. Indeed, the maximal set families are:

- $max_{\Sigma,R_s}(Emp) = \{\{Dept, Mgr\}\}$,
- $max_{\Sigma,R_s}(Dept) = \{\{Mgr\}\}$, and
- $max_{\Sigma,R_s}(Mgr) = \{\{Emp\}\}$.

The agree sets of r are the following:

- $(\emptyset, \emptyset)$,
- $(\{Dept, Mgr\}, \{Dept, Mgr\})$,
- $(\{Mgr\}, \{Mgr\})$,
- $(\{Emp\}, \{Emp, Dept\})$.

The conditions 1, 2 and 3 of Theorem 5 are satisfied.

One may expect condition 2. of Theorem 5 to read

$\forall X \in ag^s(r)(X^*_{\Sigma,R_s} = w(X))$,

but a slight change of Example 14 shows that this condition would fail to recognize some Armstrong tables.

Example 15 Let EMPLOYMENT and

$\Sigma = \{Emp \rightarrow Dept, Dept \rightarrow Mgr\}$

be as before, but let $R_s = \{Mgr\}$. The following relation r:

Emp	*Dept*	*Mgr*
Hilbert	Math	Gauss
Pythagoras	Math	Gauss
`ni`	Astronomy	Newton
`ni`	Physics	Newton
Turing	`ni`	von Neumann
Turing	`ni`	Gödel

is an Armstrong table for Σ and R_s. Indeed, the maximal set families are as before:

- $max_{\Sigma,R_s}(Emp) = \{\{Dept, Mgr\}\}$,
- $max_{\Sigma,R_s}(Dept) = \{\{Mgr\}\}$, and
- $max_{\Sigma,R_s}(Mgr) = \{\{Emp\}\}$.

The agree sets of r are:

- $(\emptyset, \emptyset)$,
- $(\{Dept, Mgr\}, \{Dept, Mgr\})$,
- $(\{Mgr\}, \{Emp, Mgr\})$,
- $(\{Emp\}, \{Emp, Dept\})$.

The conditions 1., 2. and 3. of Theorem 5 are satisfied. Note, in particular, that $(Mgr)^*_{\Sigma,R_s} = \{Mgr\}$ which is a proper subset of $w(Mgr) = \{Emp, Mgr\}$.

Finally, we illustrate Theorem 5 for the case when the input relation is not an Armstrong table for the input FD set and input NFS.

Example 16 Let the relation schema R, FD set Σ, and relation r be given as in Example 3, but let the NFS R_s be $\{Emp,Mgr\}$. Recall from Example 12 the maximal sets $\{Dept, Mgr\}$, $\{Mgr\}$ and $\{Emp\}$. The agree sets of r are:

- $(\{Dept, Mgr\},\{Dept, Mgr\})$,
- $(\{Mgr\},\{Mgr\})$,, and
- $(\emptyset, \emptyset)$.

While condition 2. of Theorem 5 is satisfied, conditions 1. and 3. are both violated. In fact, $\{Emp\} \in max_{\Sigma,R_s}(Mgr) - ag^s(r)$, and $Dept \in total(r) - R_s$.

5.3 Characterizing the satisfaction of FD sets

The next theorem generalizes the result from total relations [60] saying that a total relation satisfies a set of FDs precisely when the agree sets of the total relation are closed attribute sets.

Theorem 6 *Let R be some relation schema, let Σ be an FD set, and R_s an NFS over R. For all relations r over R it holds that r satisfies Σ and R_s if and only if $R_s \subseteq total(r)$ and $\forall X \in ag^s(r)(X^*_{\Sigma,R_s} \subseteq w(X))$ holds.*

Example 17 Example 16 shows that the relation from Example 3 satisfies the FD set Σ and NFS R_s from Example 3.

6 Computation of Armstrong tables

Following Sect. 4, we need to ask whether there is a reasonable class of FDs in the presence of an NFS that does enjoy Armstrong tables. We can give an affirmative answer: the focus on standard FDs guarantees the existence of Armstrong tables in the presence of an NFS. Instead of showing the mere existence of Armstrong tables, we establish an algorithm that computes an Armstrong table for an arbitrarily given relation schema, an arbitrarily given set of standard FDs and an arbitrarily given NFS over the relation schema.

Our refinement of the notion of an agree set in Definition 1 enables us to utilize a similar idea that Mannila and Räihä used to compute Armstrong relations for sets of FDs over total relations [96]. We specify tuples that have *strong* agree sets on the maximal sets of the attributes. In addition, these tuples are total and unique on the null-free subschema. Before we can present the computation of these Armstrong tables, we establish an algorithm for computing the families of maximal sets.

Lemma 1 *Let R be a relation schema, R_s a null-free subschema over R, and $\Sigma = \Sigma' \cup \{X \to A\}$ a set of standard FDs over R. For $WC \subseteq R$, it takes $\mathcal{O}(|R| \times ||\Sigma||)$ time to test whether $W \in max_{\Sigma,R_s}(C)$.*

Proof Using the NFSclosure-algorithm, $C \notin W^*_{\Sigma,R_s}$ can be checked in time $\mathcal{O}(||\Sigma||)$, and $C \in (WB)^*_{\Sigma,R_s}$ for all $B \in R - W$ can be checked in time $\mathcal{O}(|R| \times ||\Sigma||)$. □

We use $mtest(W, C, R, R_s, \Sigma)$ to denote the test if $W \in max_{\Sigma,R_s}(C)$ from Lemma 1. The maximal sets for R with respect to Σ and R_s can be computed by testing all subsets of R. This, however, will hardly be efficient. The following result establishes an iterative approach for computing the maximal sets for R with respect to Σ and R_s. The algorithm starts with the maximal sets for R with respect to an empty FD set in the presence of R_s, and then adds the FDs of Σ one by one while monitoring the resulting changes to the family of maximal sets.

Theorem 7 *Let R be a relation schema, R_s a null-free subschema over R, and $\Sigma = \Sigma' \cup \{X \to A\}$ a set of standard functional dependencies over R. For $C \in R$ let $V \in max_{\Sigma,R_s}(C)$. Then $V \in max_{\Sigma',R_s}(C)$ or $(C = A$ or $A \in R_s)$ holds and there is some $B \in X - V$ such that*

(i) *$VB \in max_{\Sigma',R_s}(C)$, if $X \not\subseteq R_s$, or*
(ii) *$V = W \cap Z$ for some $W \in max_{\Sigma',R_s}(C)$ and some $Z \in max_{\Sigma',R_s}(B)$.*

Theorem 7 extends Theorem 4 in [96, page 137] from the special case where $R_s = R$ to an arbitrary NFS. Indeed, for total relations case i) of Theorem 7 cannot occur, and the condition $X \subseteq R_s$ and $(C = A$ or $A \in R_s)$ is always satisfied. That is, Theorem 7 becomes Theorem 4 in [96, page 137] for the special case of total relations. Based on Theorem 7, we will now establish an algorithm for computing the maximal set families for a given relation schema R, a given set Σ of standard FDs and a given NFS R_s over R. Recall that $mtest(W, C, R, R_s, \Sigma)$ denotes the test whether $W \in max_{\Sigma,R_s}(C)$ from Lemma 1.

Algorithm 8 (Maximal set computation)

```
Input: relation schema $R$, a set $\Sigma$ of standard FDs and an
    NFS $R_s$ over $R$
Output: sets $max(C)$ of maximal sets for all $C \in R$
Method:
(A1) for all $C \in R$ let $max(C) := \{R - C\}$;
(A2) $\Theta := \emptyset$;
     for all $X \to A \in \Sigma$ do
       $\Theta := \Theta \cup \{X \to A\}$;
```

```
        for all $C \in R$ where ($C = A$ or $A \in R_s$) do
          $nmax(C) := max(C)$;
          for all $W \in max(C)$ do
            if not $mtest(W, C, R, R_s, \Theta)$ then
              $nmax(C) := nmax(C) - \{W\}$;
              for all $B \in X$ do
                if $B \notin R_s$ then
                  $nmax(C) := nmax(C) \cup \{W - B\}$;
                endif;
                for all $Z \in max(B)$ do
                  if $mtest(W \cap Z, C, R, R_s, \Theta)$ then
                    $nmax(C) := nmax(C) \cup \{W \cap Z\}$;
                  endif;
                enddo;
              enddo;
            endif;
          enddo;
        enddo;
        for all $C \in R$ let $max(C) := nmax(C)$;
      enddo;
(A3) for all $C \in R$ let
        $max(C) := \{W \mid W \in max(C)$ and $W \neq \emptyset\}$;
(A4) return $max(C)$ for all $C \in R$;
```

Theorem 7 shows that Algorithm 8 is correct.

Theorem 9 *Algorithm 8, on input (R, Σ, R_s), computes the families $max_{\Sigma,R_s}(A)$ for every $A \in R$.*

The following example illustrates that the evolution of maximal sets is conceptually different from the case of total relations, in particular for the case of a new FD whose LHS is not a subset of R_s.

Example 18 Let $R = KLM$, $\Sigma' = \{K \to L, L \to M\}$, and $\Sigma = \Sigma' \cup \{M \to K\}$. Assuming that $L \notin R_s$ we find $max_{\Sigma',R_s}(K) = \{LM\}$, $max_{\Sigma',R_s}(L) = \{M\}$, and $max_{\Sigma',R_s}(M) = \{K\}$. The remaining families of maximal sets depend on the precise choice of R_s. If $R_s = \emptyset$ we have $max_{\Sigma,R_s}(K) = \{L\}$ and $max_{\Sigma,R_s}(L) = \{M\}$. This is case *i)* in Theorem 7. It is important to note that $max_{\Sigma,R_s}(K)$ is not an intersection of maximal sets for R with respect to Σ' and R_s. If $R_s = M$ we have $max_{\Sigma,R_s}(K) = \emptyset$ and $max_{\Sigma,R_s}(L) = \{M\}$. This is the case of Theorem 7 when a new dependency is added whose RHS is not declared null-free. Note that only members of $max_{\Sigma',R_s}(K)$ can change as K is the attribute on the RHS of the new dependency. Finally, if $R_s = KM$, we have $max_{\Sigma,R_s}(K) = \emptyset$ and $max_{\Sigma,R_s}(L) = \emptyset$. This is the case of Theorem 7 where a new dependency is added whose RHS is null-free. Only this special case can be regarded as similar to [96, Theorem 4].

The following algorithm computes an Armstrong table for an arbitrary set Σ of standard FDs and an NFS R_s. Let A_0 denote an arbitrary fixed attribute of the underlying relation schema R.

Algorithm 10 (Armstrong table computation)

```
Input:   relation schema $R$, a set $\Sigma$ of standard FDs and an
         NFS $R_s$ over $R$
Output:  Armstrong table $r$ for $\Sigma$ and $R_s$
Method:  let $c_{A,1}, c_{A,2}, \ldots \in dom(A)$ be distinct
(A0) for all $A \in R$ compute $max(A)$ (Alg. 8);
(A1) $r := \emptyset$; $i := 1$;
(A2) for all $X \in max(R)$ do
(A3)     $Z := \{A \in R \mid X \in max(A)\}$;
(A4)     $r := r \cup \{t_i, t_{i+1}\}$ where $\forall A \in R$
           $t_i(A) := \begin{cases} c_{A,i} & \text{, if } A \in XZR_s \\ \texttt{ni} & \text{, else} \end{cases}$ ; and
           $t_{i+1}(A) := \begin{cases} c_{A,i} & \text{, if } A \in X \\ c_{A,i+1} & \text{, if } A \in Z(R_s - X) \\ \texttt{ni} & \text{, else} \end{cases}$ ;
(A5)     $i := i + 2$;
(A6) enddo;
(A7) $total(r) := \{A \in R \mid \forall t \in r(t[A] \neq \texttt{ni})\}$;
       if $total(r) - R_s \neq \emptyset$, then
         if $R_s = \emptyset$ and $total(r) = R$ and $|R| > 1$, then
           return $r := r \cup \{t_i, t_{i+1}\}$ where $\forall A \in R$
           $t_i(A) := \begin{cases} c_{A,i} & \text{, if } A = A_0 \in R \\ \texttt{ni} & \text{, else} \end{cases}$ ; and
           $t_{i+1}(A) := \begin{cases} \texttt{ni} & \text{, if } A = A_0 \in R \\ c_{A,i+1} & \text{, else} \end{cases}$ ;
         else return $r := r \cup \{t_i\}$ where $\forall A \in R$
           $t_i(A) := \begin{cases} \texttt{ni} & \text{, if } A \in total(r) - R_s \\ c_{A,i} & \text{, else} \end{cases}$
         endif;
       else return $r$ endif;
```

The main construction is achieved in steps $(A2) - (A6)$. Step $(A7)$ guarantees that the output relation is total on exactly those attributes that belong to R_s, and that the relation is subsumption-free. For the soundness proof of Algorithm 10, we utilize Atzeni and Morfuni's axiomatization [6].

Theorem 11 *Algorithm 10, on input (R, Σ, R_s), computes an Armstrong table for Σ and R_s.*

Example 19 Let EMPLOYMENT={*Emp*, *Dept*, *Mgr*},

$\Sigma = \{Emp \to Dept, Dept \to Mgr\}$

and $R_s = \{Emp, Mgr\}$ be as in Example 2. The following table illustrates the evolution of the maximal set families for the attributes of EMPLOYMENT using Algorithm 8 on input (EMPLOYMENT, Σ, R_s) when the first FD considered is $\sigma = Dept \to Mgr$:

A	$max_{\emptyset,R_s}(A)$	$max_{\{\sigma\},R_s}(A)$	$max_{\Sigma,R_s}(A)$
Emp	{{*Dept*,*Mgr*}}	{{*Dept*,*Mgr*}}	{{*Dept*,*Mgr*}}
Dept	{{*Emp*,*Mgr*}}	{{*Emp*,*Mgr*}}	{{*Mgr*}}
Mgr	{{*Emp*,*Dept*}}	{{*Emp*}}	{{*Emp*}}

.

Based on the elements of max_{Σ,R_s}, Algorithm 10 would compute the following Armstrong table

Emp	*Dept*	*Mgr*
$c_{Emp,1}$	$c_{Dept,1}$	$c_{Mgr,1}$
$c_{Emp,2}$	$c_{Dept,1}$	$c_{Mgr,1}$
$c_{Emp,3}$	$c_{Dept,3}$	$c_{Mgr,3}$
$c_{Emp,4}$	$c_{Dept,4}$	$c_{Mgr,3}$
$c_{Emp,5}$	ni	$c_{Mgr,5}$
$c_{Emp,5}$	ni	$c_{Mgr,6}$

for Σ and R_s.

Finally, we state some consequences of the results we have established in this section.

Corollary 2 *The class of standard FDs enjoys Armstrong tables in the presence of an NFS.*

Algorithm 10 can also be used to compute Armstrong relations for Lien's class of standard FDs [89]. In that case, we simply set the null-free subschema to $R_s = \emptyset$, but step $(A7)$ is unnecessary since we do not need to consider any NFS.

Corollary 3 *Algorithm 10, on input $(R, \Sigma, \emptyset)$, computes an Armstrong relation with respect to Σ.*

Corollary 4 *Lien's class of standard FDs enjoys Armstrong relations.*

Note that Algorithm 10 can also be used to compute Armstrong relations for the class of standard FDs over total relations.

7 Complexity results

We show that the gain in generality we have established for the toolbox of Armstrong relations does not result in a loss of efficiency when compared to the special case of total relations. Firstly, the problem of finding an Armstrong table for a given set of standard FDs and a given NFS remains precisely exponential in the size of the input, as was the case for total relations [9]. Nevertheless, our algorithm for computing Armstrong tables remains quite conservative in the sense that the size of the output is at most quadratic in the size of the best output possible, similar to the special case of total relations [96]. Next, we examine the most concise way of representing the information inherent in an FD set. We conclude that already for the special case of total relations neither the representation in form of an FD set or the representation in form of an Armstrong table strictly dominates the other. Finally, we show that the problem of deciding whether there is some Codd key with at most k attributes that is implied by a given set of FDs and an NFS with k attributes is NP-complete for the representation either as FD set or an Armstrong table thereof. Hence, the complexity remains the same as in the special case of total relations [9,92].

7.1 The time-complexity to find Armstrong tables

The user-friendly representation of an FD set and an NFS in form of an Armstrong table comes, in general, at a price. In fact, the number of tuples in a minimum-sized Armstrong table can be exponential in the number of attributes. Due to this result, we cannot design an algorithm for generating Armstrong tables in polynomial time in the worst case. The next result shows that the number of attribute sets maximal for Σ and R_s is a lower bound for the number of agree sets found in any Armstrong table for Σ and R_s. A similar result holds for the special case of total relations [9].

Proposition 1 *Let Σ be a set of standard FDs, let R_s be some NFS over some relation schema R, and let r be an Armstrong table for Σ and R_s. Then $|max_{\Sigma,R_s}(R)| \leq |ag(r)| \leq \binom{|r|}{2}$.*

We recall what we mean by *precisely exponential* [9]. Firstly, it means that there is an algorithm for computing an Armstrong table, given a set Σ of standard FDs and an NFS R_s, where the running time of the algorithm is exponential in the number of attributes. Secondly, it means that there is a set Σ of standard FDs and an NFS R_s in which the number of tuples in each minimum-sized Armstrong table for Σ and R_s is exponential—thus, an exponential amount of time is required in this case simply to write down the relation.

Proposition 2 *The complexity of finding an Armstrong table, given a set of standard functional dependencies and a null-free subschema, is precisely exponential in the number of attributes.*

7.2 The size of minimum-sized Armstrong tables

Despite the general worst-case exponential complexity in the number of attributes, Algorithm 10 is a fairly simple algorithm for generating Armstrong tables that is, as we show now, quite conservative in its use of time.

Let the size of an Armstrong table be defined as the number of tuples that it contains. In practice, the most appealing Armstrong table for an FD set Σ should be of minimum size. The reason is that a small number of tuples is easier to comprehend for humans. Therefore, it is a practical question to ask how many tuples a minimum-sized Armstrong table requires. An Armstrong table r for Σ and R_s is said to be *minimum-sized* if there is no Armstrong table r' for Σ and R_s such that $|r'| < |r|$. That is, for a minimum-sized Armstrong table for Σ and R_s, there is no Armstrong table for Σ and R_s with a smaller number of tuples.

Proposition 3 *Let Σ be a set of standard FDs, let R_s be some NFS over some relation schema R, and let r be a minimum-sized Armstrong table for Σ and R_s. Then*

$$\frac{\sqrt{1+8\cdot|max_{\Sigma,R_s}(R)|}}{2} \leq |r| \leq 2\times|max_{\Sigma,R_s}(R)|+1.$$

We conclude that Algorithm 10 always computes an Armstrong table of reasonably small size.

Corollary 5 *On input (R, Σ, R_s), Algorithm 10 computes an Armstrong table for Σ and R_s whose size is at most quadratic in the size of a minimum-sized Armstrong table for Σ and R_s.*

There are also instances for which Algorithm 1 computes a minimum-sized Armstrong table. For each $n \geq 3$, such an instance is given by $R_n = A_1 \cdots A_n$, $R_s = \emptyset$, and $\Sigma = \bigcup_{i=1}^{n-1}\{A_i \rightarrow A_{i+1}\} \cup \{A_n \rightarrow A_1\}$.

Theorem 12 *There is some relation schema R, some NFS R_s and some standard FD set Σ over R such that Algorithm 10 computes a minimum-sized Armstrong table for Σ and R_s.*

7.3 The size of representations

We show that, in general, there is no most concise way of representing the information inherent in a set of standard FDs and a null-free subschema. We have already seen a case where the representation using Armstrong tables can be exponentially larger than the best equivalent FD set, see Proposition 2.

Corollary 6 *There is some standard FD set Σ and an NFS R_s such that Σ has size $\mathcal{O}(n)$, and the size of a minimum-sized Armstrong table for Σ and R_s is $\mathcal{O}(2^{n/2})$.*

The following theorem shows that in other cases, the representation using Armstrong tables can be exponentially smaller than the best representation using FD sets. Extending Maier's notion of an optimal cover from total relations [93], for an FD set Σ and an NFS R_s we call an FD set Σ' an *optimal cover* of Σ with respect to R_s if

- Σ' is a cover of Σ with respect to R_s, i.e., for every FD $\sigma \in \Sigma$ we have $\Sigma' \models_{R_s} \sigma$; and for every FD $\sigma' \in \Sigma'$ we have $\Sigma \models_{R_s} \sigma'$; and
- there is no cover Σ'' of Σ with respect to R_s such that Σ'' contains fewer symbol occurrences than Σ' (repeated symbol occurrences are counted as many times as they occur).

Let $R = A_1 B_1 \cdots A_n B_n C$, $R_s = R$ and

$$\Sigma = \{X_1, \ldots, X_n \rightarrow C \mid \forall i = 1, \ldots, n(X_i \in \{A_i, B_i\})\}.$$

Then, Σ is its own optimal cover with respect to R_s, but there is an Armstrong table for Σ and R_s where the number of tuples is in $\mathcal{O}(n)$.

Theorem 13 *There is some relation schema R, some NFS R_s and some standard FD set Σ over R such that there is an Armstrong table for Σ and R_s where the number of tuples is in $\mathcal{O}(n)$, and the optimal cover of Σ with respect to R_s has size $\mathcal{O}(2^n)$.*

We can see that the representation in form of an Armstrong table can offer tremendous space savings over the representation as an FD set, and vice versa.

7.4 The time-complexity to find a Codd key

Finally, it follows easily that the well-known problem [9, 92] of deciding whether there is a key of size at most k is also *NP*-complete when functional dependencies and null-free subschemata are given. An input to the problem can be either a relation schema R, a set Σ of FDs and an NFS R_s over R, and a non-negative integer $k \leq |R_s|$, or a relation schema R, an Armstrong table r over R for some FD set Σ and some NFS R_s, and a non-negative integer $k \leq |R_s|$. For the first type of input, the problem is to decide whether there is a Codd key $Codd(X)$ with $|X| \leq k$ such that $\Sigma \models_{R_s} Codd(X)$. For the second type of input, the problem is to decide whether there is a Codd key $Codd(X)$ with $|X| \leq k$ such that r satisfies $Codd(X)$. We say that we are deciding the problem if there is a Codd key of size at most k for either type of input. For the hardness part of the next result, we simply choose R_s to be R [9].

Proposition 4 *For either type of input, the problem of deciding whether there is a Codd key of size at most k is NP-complete.*

8 Extensions of our results

In this section, we establish some extensions of our results. We analyze the impact of lifting the restriction to subsumption-free relations, and study Armstrong tables for standard functional dependencies with respect to the class of standard and non-standard functional dependencies.

8.1 Subsumption in SQL tables

So far, we have imposed the following restriction: No relation in the database shall contain two tuples t_1 and t_2 such that t_1 subsumes t_2. The reason was to keep consistency with previous work in this area [6,89,91,128]. In this subsection, we analyze the impact on the validity of our results when lifting the restriction.

In general, all the results on Armstrong tables we have established for the class of functional dependencies remain valid. Furthermore, if a Codd key $Codd(X)$ has been specified on the schema, then all the relations over the schema will be subsumption-free. For the remainder of the discussion, we therefore look at schemata where no Codd key has been specified. In this case, Theorem 2 fails. In fact, there are now relations over R that satisfy the FD $X \rightarrow R$, but still violate the uniqueness constraint $unique(X)$. As a consequence, there is a real interest in studying the combined class of uniqueness constraints and functional dependencies in the presence of an NFS. We exclude the non-standard uniqueness constraint $unique(\emptyset)$ from our investigation since it would force our relations to contain at most one tuple. Note that for a set Σ of uniqueness constraints and FDs, and an NFS R_s over R, the following holds: $\Sigma \models_{R_s} Codd(X)$ if and only if $\Sigma \models_{R_s} unique(X)$ and $X \subseteq R_s$. Hence, over relations in which subsumption is allowed the combined class of uniqueness constraints and FDs in the presence of an NFS subsumes the class of Codd keys.

For the remainder of this discussion, we analyze how to compute Armstrong tables for the combined class of uniqueness constraints and FDs in the presence of an NFS. Given a set $\Sigma = \Sigma_{\text{unique}} \cup \Sigma_{\text{FD}}$ of standard uniqueness constraints and standard FDs over relation schema R, we first use Algorithm 10, exclusive of step (A7), to compute an Armstrong table r' for the set $\Sigma^{\text{FD}}_{\text{unique}} \cup \Sigma_{\text{FD}}$ of standard FDs. Here,

$$\Sigma^{\text{FD}}_{\text{unique}} := \{X \rightarrow R \mid unique(X) \in \Sigma_{\text{unique}}\}$$

consists of the FDs implied by the uniqueness constraints in Σ_{unique}. However, we need to add further tuples to r' in order to obtain an Armstrong table for Σ. In fact, we need to ensure additionally that all uniqueness constraints $unique(X)$ over R are violated that are not implied by Σ. If Σ does not imply the FD $X \rightarrow R$, then r' already violates $unique(X)$. Therefore, it suffices to look at the following family of attribute sets:

$$\begin{aligned} dup_{\Sigma,R_s}(R) = \{X \subseteq R \mid \Sigma \models_{R_s} X \rightarrow R \ \wedge \\ \Sigma \not\models_{R_s} unique(X) \ \wedge \\ \forall A \in R - X (\Sigma \models_{R_s} unique(XA))\}. \end{aligned}$$

To compute $dup_{\Sigma,R_s}(R)$, we generate the hyper-graph $\mathcal{H}=(V,E)$ with vertex set $V = R$ and the set

$$E = \{K - R_s \mid unique(K) \in \Sigma_{\text{unique}}\}$$

as hyper-edges. From this, we obtain $dup_{\Sigma,R_s}(R)$ as

$$\begin{aligned} dup_{\Sigma,R_s}(R) = \{R - X \mid X \in Tr(\mathcal{H}) \ \wedge \\ \forall M \in max_{\Sigma,R_s}(R)(R - X \not\subseteq M)\}. \end{aligned}$$

where $Tr(\mathcal{H})$ denotes the minimal transversals of the hypergraph $\mathcal{H}$ [62]. Algorithm 14 extends Algorithm 10 in order to compute an Armstrong table for a set of uniqueness constraints and FDs in the presence of an NFS, considering that relations are not necessarily subsumption-free. In particular, step (A2) adds to the table tuples that strongly agree on the attribute sets in $dup_{\Sigma,R_s}(R)$. This ensures that the resulting table violates all uniqueness constraints not implied by Σ. Note that for all $X \in dup_{\Sigma,R_s}(R)$, we have $R_s \subseteq X$. Step (A3) is a simplification of step (A7) from Algorithm 10 since we do not need to ensure any longer that the resulting Armstrong table is subsumption-free.

Algorithm 14 (Armstrong tables revised)

```
Input:   relation schema $R$, set $\Sigma_{\text{unique}} \cup \Sigma_{\text{FD}}$ of standard
    uniqueness constraints and FDs, and an NFS $R_s$ over $R$
Output:  Armstrong table $r$ for $\Sigma_{\text{unique}} \cup \Sigma_{\text{FD}}$ and $R_s$
Method:  let $c_{A,1}, c_{A,2}, \ldots \in dom(A)$ be distinct
(A1)  let $r$ be the output of Algorithm 10 from steps
      (A0)-(A6) on input $(R, \Sigma^{\text{FD}}_{\text{unique}} \cup \Sigma_{\text{FD}}, R_s)$;
(A2)  if $\neg\exists X(unique(X) \in \Sigma_{\text{unique}} \wedge X \subseteq R_s)$ then
        for all $X \in dup_{\Sigma,R_s}(R)$ do
          $r := r \cup \{t_i, t_{i+1}\}$ where $\forall A \in R$
            $t_i(A) = t_{i+1}(A) := \begin{cases} c_{A,i} & \text{, if } A \in X \\ \texttt{ni} & \text{, else} \end{cases}$;
          $i := i + 2$;
        enddo;
      endif;
(A3)  $total(r) := \{A \in R \mid \forall t \in r(t[A] \neq \texttt{ni})\}$;
      if $total(r) - R_s \neq \emptyset$, then
        return $r := r \cup \{t_i\}$ where $\forall A \in R$
          $t_i(A) := \begin{cases} \texttt{ni} & \text{, if } A \in total(r) - R_s \\ c_{A,i} & \text{, else} \end{cases}$
      else return $r$ endif;
```

Example 20 Consider again the relation schema EMPLOYMENT with attribute set *Emp*, *Dept* and *Mgr*, FD set Σ_{FD} that consists of $Emp \rightarrow Dept$ and $Dept \rightarrow Mgr$, and NFS R_s consisting of *Emp* and *Mgr*. In particular, let Σ_{unique} be empty. Then the relation

Emp	*Dept*	*Mgr*
Hilbert	Math	Gauss
Hilbert	Math	Gauss
Pythagoras	Math	Gauss
Einstein	Physics	Gauss
Turing	ni	von Neumann
Turing	ni	Gödel

is an Armstrong table for $\Sigma_{unique} \cup \Sigma_{FD}$ in the presence of R_s. Here, $dup_{\Sigma,R_s}(R) = \{R\}$ and the first two tuples are duplicates illustrating that no uniqueness constraint is considered meaningful with the present design.

We now consider Beeri and Bernstein's famous example that was originally used to show that dependency-preserving Boyce-Codd Normal Form decompositions cannot always be obtained in the relational model of data [7]. Here, we apply our toolbox of Armstrong tables to the relational approximation of the application domain to derive a concise SQL table definition.

Example 21 Suppose a team of data engineers has decided to use the three attributes *Address*, *City*, and *ZIP* as part of a contact address management system. They have identified the set Σ_{FD} with $Address, City \to ZIP$, and $ZIP \to City$ as a first approximation for capturing the semantics of the underlying application domain. (This is the example from [7].) Suppose that Σ_{unique} and the NFS R_s remain empty. Using our toolbox the team produces

Address	*City*	*ZIP*
03 Hudson St	Manhattan	10001
03 Hudson St	Manhattan	10001
70 King St	Manhattan	10001
70 King St	San Francisco	94107
ni	San Francisco	94129
15 Maxwell St	ni	ni

as an Armstrong table for $\Sigma_{unique} \cup \Sigma_{FD}$ in the presence of R_s. The domain experts inspect the table. They note from the first two tuples that the FD $Address, City \to ZIP$ should be replaced by the stronger uniqueness constraint *unique*(*Address*,*City*). Hence, the revised constraint set becomes: $\Sigma_{\text{unique}} = \{unique(Address, City)\}$, $\Sigma_{\text{FD}} = \{ZIP \to City\}$, and R_s is the empty set. For these requirements, they produce

Address	*City*	*ZIP*
03 Hudson St	Manhattan	10001
70 King St	Manhattan	10001
70 King St	San Francisco	94107
ni	San Francisco	ni
15 Maxwell St	ni	60609
15 Maxwell St	ni	60609

as an Armstrong table for $\Sigma_{unique} \cup \Sigma_{FD}$ in the presence of R_s. In particular, $dup_{\Sigma,R_s}(R)$ contains the set $\{Address, ZIP\}$. The domain experts are quite happy with the table, but are confused about the last two rows. After some discussion, there is consensus that there is no reason to allow different rows with the same total values on *Address* and *ZIP*. Indeed, the data engineers realize that they have not captured the uniqueness constraint $unique(Address, ZIP)$. (It is important to note here that in the relational model the FD $ZIP \to City$ implies the key {*Address*,*ZIP*}, but over SQL tables the two constraints *unique*(*Address*,*City*) and $ZIP \to City$ do not imply *unique*(*Address*,*ZIP*).) For this reason, the team revises the set of constraints again: Σ_{unique} consist of $unique(Address, City)$ and $unique(Address, ZIP)$, and $\Sigma_{\text{FD}} = \{ZIP \to City\}$. R_s still remains empty. Then, they produce

Address	*City*	*ZIP*
03 Hudson St	Manhattan	10001
70 King St	Manhattan	10001
70 King St	San Francisco	94107
ni	San Francisco	ni
15 Maxwell St	ni	60609

as an Armstrong table for $\Sigma_{unique} \cup \Sigma_{FD}$ in the presence of R_s. Here, $dup_{\Sigma,R_s}(R) = \emptyset$. The first two tuples ensure that the uniqueness constraint $unique(City, ZIP)$ is violated. The domain experts note now that there is a need to always have precise information in the *Address* and *ZIP* columns. As a consequence, the data engineers decide to include both column headers in the null-free subschema R_s. This is the same as having the following constraints: *Codd*(*Address*,*ZIP*), *unique*(*Address*,*City*) and FD $ZIP \to City$. The table

Address	*City*	*ZIP*
03 Hudson St	Manhattan	10001
70 King St	Manhattan	10001
70 King St	San Francisco	94107
35 Lincoln Blvd	San Francisco	94129
15 Maxwell St	ni	60609

is an Armstrong table for this design choice. One possible SQL table definition is the following:

```
CREATE  TABLE CONTACT (
  Address VARCHAR,
  City VARCHAR,
  ZIP INT,
  UNIQUE(Address,City),
  PRIMARY KEY(Address,ZIP),
  CHECK(Q = 0));
```

The state assertion is based on the following query Q:

```
SELECT COUNT(*)
FROM CONTACT c1
WHERE c1.ZIP IN (
  SELECT ZIP
  FROM CONTACT c2
  WHERE c1.ZIP=c2.ZIP
    AND (c1.City <> c2.City
    OR (c1.City IS NULL AND c2.City IS NOT NULL)
    OR (c1.City IS NOT NULL AND c2.City IS NULL)));
```

and can be enforced on the data or middle tier.

8.2 Armstrong tables and non-standard FDs

One may also investigate the properties of Armstrong tables for a set of standard FDs and an NFS with respect to the class of all FDs. More formally, a relation r is said to be an Armstrong table for a set Σ of standard FDs and an NFS R_s *in the world of all FDs*, if $total(r) = R_s$ and the following holds for all standard and non-standard FDs $\sigma : r$ satisfies σ if and only if $\Sigma \models_{R_s} \sigma$. Since any set of standard FDs does not imply any non-trivial non-standard FD, this definition can be simplified as follows. A relation r is an Armstrong table for a set Σ of standard FDs and an NFS R_s in the world of all FDs, if $total(r) = R_s$, r violates every non-trivial non-standard FD, and the following holds for all standard FDs $\sigma : r$ satisfies σ if and only if $\Sigma \models_{R_s} \sigma$.

Every Armstrong table r for a set Σ of standard FDs and an NFS R_s can be easily modified to obtain an Armstrong table for Σ and R_s in the world of all FDs: just add to r an additional tuple with previously unused domain values different from ni. An exception is the special case where the underlying relation schema consists of only one attribute A and the NFS $R_s = \emptyset$. In this case, no subsumption-free relation can simultaneously violate the NFS R_s and the non-trivial, non-standard FD $\emptyset \to A$. Consequently, no Armstrong table exists for the empty FD set Σ and the NFS $R_s = \emptyset$ in the world of all FDs, if the underlying relation schema consists of only one attribute. Note, however, that the singleton consisting of the tuple $t = \texttt{ni}$ is an Armstrong table for the empty FD set Σ and the NFS $R_s = \emptyset$ in the "world of all standard FDs". Hence, there are relations that are Armstrong tables in the world of all standard FDs, but not an Armstrong table in the world of all FDs. Note that every relation that is an Armstrong table in the world of all FDs is also an Armstrong table in the world of all standard FDs.

How do the results for standard FD sets from the previous sections change in the world of all FDs? Regarding the notion of maximal sets in Definition 3, it is intuitive to allow also empty attribute sets to be maximal for any attribute. The characterization of Armstrong tables in Theorem 4 carries over to the world of all FDs, if the first condition is true for all attribute sets X including the empty one. The characterization of Theorem 5 carries over to the world of all FDs as it is. Theorem 6 carries over with no change since it is not concerned with the violation of any non-standard FDs. The next example illustrates these points.

Example 22 Consider $R = \{Dept, Mgr\}$ with standard FD set $\Sigma = \{Dept \to Mgr\}$ and NFS $R_s = R$. The relation r

Dept	*Mgr*
Math	Gauss
Physics	Gauss

is an Armstrong table for Σ and R_s, but not an Armstrong table for Σ and R_s in the world of all FDs: the non-standard FD $\emptyset \to Mgr$ is not implied by Σ and R_s, but satisfied by r. Indeed, the empty attribute set X violates the first condition of Theorem 4, and the first condition of Theorem 5 is violated since the empty attribute set is an element of $max_{\Sigma,R_s}(Mgr) - ag^s(r)$. Adding $t =$ (CS, von Neumann) to r results in an Armstrong table for Σ and R_s in the world of all FDs.

Table 2 Armstrong table for Σ and ORDER$_s$

Order ♯	Product ♯	Description	Qty	Total
10723	389	Microwave	50	25000
21834	389	Microwave	50	25000
21834	490	Microwave	50	25000
21834	501	Fridge	50	25000
21834	612	Fridge	75	25000
32945	612	Fridge	100	25000
32945	723	Fridge	100	30000
43056	834	Oven	ni	35000
43056	834	Oven	ni	40000
54167	945	ni	200	ni

For the computation of the maximal sets, the only change in Algorithm 8 is concerned with step (A3) where we simply remove the condition that W must not be the empty set. Regarding the computation of Armstrong tables, Algorithm 10 does not require any changes. However, the special case where $|R| = 1$ and $R_s = \emptyset$ should be excluded from the set of all possible inputs to the algorithm since no Armstrong table exists in the world of all FDs, as mentioned before. Corollaries 3 and 4, as well as the results from Sect. 7 carry over to the world of all FDs.

9 Impact and applications

In this section, we demonstrate the potential impact of our results on various database applications.

An example domain. Let us assume that in developing an information system for some manufacturer of electrical goods, we identify the processing of orders by retail sellers as a domain of interest. In particular, we define a relation schema ORDER that consists of the attributes *Order♯*, *Product♯*, *Description*, *Qty*, and *Total*. These show for an order (identified by its order number *Order♯*), a product in that order (identified by its unique product number *Product♯*), a description *Description* of that product, the quantity *Qty* of that product in that order, and the total value *Total* (in some fixed currency) of that product in that order.

The data engineers of our information system have identified the following FD set Σ to be meaningful: $Order\sharp, Product\sharp \to Qty$, $Product\sharp \to Description$, and $Product\sharp, Qty \to Total$. The team of data engineers agrees on the NFS ORDER$_s = \{Order\sharp, Product\sharp\}$.

Before the implementation, the team validates their design using the Armstrong table for Σ and ORDER$_s$ from Table 2. The engineers observe that the product with *Product♯* 612 occurs with the same total value *Total* of 25000, but with the different quantities *Qty* of 75 and 100. After consultation with the domain experts, the team responds by including the additional meaningful FD $Product\sharp, Total \rightarrow Qty$ in Σ. The team further notices that *Codd*(*Order*♯,*Product*♯) is not implied by their design choice: the product with *Product*♯ 834 in the order with *Order*♯ 43056 has different total values of 35000 and 40000. After some discussion, the team responds by re-defining the NFS to $\text{ORDER}_s = \{Order\sharp, Product\sharp, Qty\}$. The inspection of the Armstrong table has resulted in the recognition of additional meaningful business rules.

Efficient processing of updates. The main driver of database normalization is to avoid data redundancy to guarantee efficient updates. The Boyce-Codd Normal Form (BCNF) enforces a syntactic condition on relation schemata that eliminates data redundancy in terms of FDs [8,12,120]. We assume familiarity with the definition of such terms as BCNF, lossless and dependency-preserving decomposition [1]. Our schema ORDER is not in BCNF with respect to the FD set Σ containing

- $Order\sharp, Product\sharp \rightarrow Qty$,
- $Product\sharp \rightarrow Description$,
- $Product\sharp, Qty \rightarrow Total$, and
- $Product\sharp, Total \rightarrow Qty$,

and the NFS $\text{ORDER}_s = \{Order\sharp, Product\sharp, Qty\}$. Following standard techniques [1], we decompose ORDER into PRODUCT={*Product*♯,*Description*} with the FD set $\Sigma_1 = \{Product\sharp \rightarrow Description\}$ and $\text{PRODUCT}_s = \{Product\sharp\}$, into TOTAL={*Product*♯,*Qty*,*Total*} with the FD set Σ_2 containing

$$Product\sharp, Qty \rightarrow Total \quad \text{and} \quad Product\sharp, Total \rightarrow Qty,$$

and $\text{TOTAL}_s = \{Product\sharp, Qty\}$, and QTY={*Order*♯, *Product*♯,*Qty*} with FD set $\Sigma_3 = \{Order\sharp, Product\sharp \rightarrow Qty\}$, and $\text{QTY}_s = \{Order\sharp, Product\sharp, Qty\}$. The three relation schemata represent a lossless and dependency-preserving BCNF decomposition of ORDER for Σ and ORDER_s. Note that schema ORDER with the original FD set Σ and NFS $\text{ORDER}_s = \{Order\sharp, Product\sharp\}$ every BCNF decomposition is lossy or not dependency-preserving. Hence, an inspection of the Armstrong table enabled our data engineers to find a database layout that represents all of the business rules, permits efficient consistency checking, is free from data redundancies and update anomalies.

Efficient processing of queries. Besides updates, the efficient processing of database queries is also a significant task of DBMSs. Indeed, there has been a lot of research regarding the use of FDs and other constraints in semantic query optimization and query processing with SQL databases including [14,26,42,70,71,103–105,111]. We will illustrate how the use of our Armstrong tables can result in semantically optimized query re-writings. Recall that the inspection of the Armstrong table enabled the data engineers to identify the meaningful FD $Product\sharp, Total \rightarrow Qty$ and to specify *Qty* as NOT NULL. The Codd key $Codd(Order\sharp, Product\sharp)$ is implied by Σ and ORDER_s, in particular. Consider the query that retrieves all combinations of order numbers and quantities associated with the same product and total value. A naive implementation of this query is

```
SELECT   ORDER'.Order♯, ORDER.Qty
FROM     ORDER, ORDER AS ORDER'
WHERE    ORDER.Product♯=ORDER'.Product♯ AND
         ORDER.Total ♯ =ORDER'.Total ♯
```

However, since $Product\sharp, Total \rightarrow Qty$ has been correctly specified as a meaningful FD, the quantity is the same for each given combination of product and total value. Hence, the query above can be rewritten into

```
SELECT  ORDER.Order♯, ORDER.QtyFROM  ORDER
```

which is much simpler and requires no join.

Inference control is a security mechanism developed to ensure confidentiality in databases [18,19,58]. The objective is to avoid inferences of secrets by users based on their query history and their knowledge about the database. Suppose the fact that there is an order of a *Fridge* with a total value of *100000* is a business secret. Hence, the fact that the next sentence Ψ is true in the database must not be revealed to unauthorized users: $\exists X_O \exists X_P \exists X_Q \text{ORDER}(X_O, X_P, Fridge, X_Q, 100000)$.
A user may issue the queries:

- $\Phi_1 = \exists X_Q \exists X_T \text{ORDER}(73956, 1645, Fridge, X_Q, X_T)$ and
- $\Phi_2 = \exists X_D \exists X_T \text{ORDER}(73956, 1645, X_D, X_Q, 100000)$

and learn that both queries are true in the current instance, since neither Φ_1 nor Φ_2 individually reveal Ψ. The user may anticipate that the instance satisfies $Order\sharp, Product\sharp \rightarrow Description, Total$, based on the semantics of the application domain. Hence, the user may apply this FD to the answers to Φ_1 and Φ_2 to infer that $\exists X_Q \text{ORDER}(73956, 1645, Fridge, X_Q, 100000)$ is true in the database instance. This, however, would reveal the potential secret to the user. Fortunately, the data engineers were able to utilize Armstrong tables to recognize that the FD $Order\sharp, Product\sharp \rightarrow Description, Total$ is indeed a meaningful constraint for the application domain. Therefore, the security officer is able to anticipate such an inference, and distort the answers to the queries suitably. In this example, it would suffice to refuse an answer to query Φ_2 after the user has learned Φ_1.

Consequently, Armstrong tables also provide an aid that can help security officers to better understand the opportunities of potential inference attacks on future database instances, and therefore may prove invaluable in preventing such attacks successfully.

Design approaches. Our toolbox of Armstrong tables *complements* any existing approaches to schema design, such as Entity-Relationship modeling [45,117] and relational normalization [1,94]. As illustrated by Example 21, Armstrong tables provide data engineers with concise test data that helps them to consolidate their final design choice. It also helps them to communicate and justify their choice to other stakeholders of the database. It is our assumption that the inspection of Armstrong tables exposes those semantically meaningful uniqueness constraints and FDs that are not implied by the set of constraints presently considered meaningful by the data engineers. Recent studies confirm this assumption empirically, already for relations where no duplicate rows and no partial data are allowed to occur [82]. Suppose that the schema $\{Address, City, ZIP\}$ with *Codd*(*Address*,*ZIP*) and *unique*(*Address*,*City*) constitutes the current design choice. An Armstrong table for the two uniqueness constraints will expose the FD $ZIP \rightarrow City$ as a meaningful constraint that is not captured by this choice. Finally, de-normalization is often applied in practice to increase the efficiency of query processing [1]. Armstrong tables will be particularly useful in identifying the FDs exhibited by de-normalized schemata.

10 Conclusion and future work

We have investigated the existence and properties of Armstrong tables for Atzeni and Morfuni's class of FDs and NFS over relations that can contain occurrences of Zaniolo's no information null value. This covers the class of FDs defined over SQL table definitions that contain `NOT NULL` constraints and has therefore important implications for the processing of data in real database systems. In contrast to the special case of total relations, we have shown that FDs do not enjoy Armstrong tables, in general. However, the class of standard FDs and NFSs does enjoy Armstrong tables. We have given sufficient and necessary conditions for a given table to be an Armstrong table for a given set of standard FDs and a given NFS. In general, the problem of finding an Armstrong table remains precisely exponential in the number of attributes. However, we have established a provably correct algorithm that computes an Armstrong table in time that is at most quadratic in the size of a minimum-sized Armstrong table. We have demonstrated that data engineers should utilize not only an abstract representation in form of FD sets, but also the representation in form of an Armstrong table. For SQL tables that are not necessarily subsumption-free, we have extended our results to the combined class of uniqueness constraints and FDs in the presence of an NFS. In particular, our Armstrong tables can visualize the delicate interactions of these constraints that can occur in real SQL tables, but not in the relational model of data. Finally, we have illustrated that the utilization of Armstrong tables can lead to the identification of meaningful constraints, and result in database designs that facilitate efficient updates and queries, and are less prone to inference attacks. In summary, our contributions extend well-known results from total relations to SQL tables. Hence, the resulting toolbox can be applied to instances that occur in real database systems, with only small space and no time penalties when compared to the previously studied special case of total relations.

For future work, it would be interesting to investigate the properties of Armstrong tables for the combined class of functional and inclusion dependencies in the presence of a null-free subschema. Levene and Loizou have investigated the implication problem for the class of inclusion dependencies where all attributes are assumed to be `NULL` [84]. Mannila and Räihä have also developed algorithms to compute Armstrong databases for the class of functional and acyclic inclusion dependencies over total relations [97].

Fagin has shown that Armstrong relations exist for so-called *implicational dependencies* [50]. It would be a worthwhile endeavor to identify broad classes of dependencies that enjoy Armstrong relations in the presence of null values, e.g., multivalued dependencies [69,89,91].

A main observation for the areas of impact from Sect. 9 is that most of the existing theory does not apply to SQL tables. These areas include normalization [48], semantic query optimization [26,42], consistent query answering [27], and controlled query evaluation [16].

We plan to implement our concepts and algorithms to extend design aids available for total relations [35,36,96, 110]. It appears to be intuitive that data engineers find it more difficult to understand the interaction between uniqueness constraints and FDs in the presence of an NFS than that of just FDs over total relations. Hence, Armstrong tables might be of even bigger value than reported for the special case of total relations [82].

In Jagadish et al. [106] develop a user-friendly approach toward spreadsheet design. Here, functional dependencies are discovered from the given spreadsheet data. In other words, the spreadsheet data are an Armstrong relation for the set of FDs discovered. The FDs are then used to recommend auto-completions for updates and to warn users about potential data entry errors. Based on the users' reply to these warnings, the functional dependencies are maintained incrementally. A user study showed that these guidance features do improve usability [106]. However, spreadsheet data cannot be assumed to be complete in practice. It would therefore be interesting to extend this approach to partial information, for example under the *no information* interpretation of null values.

It is also an interesting problem to study the properties of Armstrong data trees for FDs in the context of XML. The results of our article should provide valuable information to learn more about the properties of Armstrong data trees for several classes of XML FDs.

Other directions include the problem of dependency inference [98] or data cleaning [54], but also the investigation of extremal problems [39,47] in the context of partial relations. Finally, all these problems should also be investigated for other interpretations of null values [30,63–65,95].

Acknowledgments This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand. Sven Hartmann is supported by a research grant of the Alfried Krupp von Bohlen and Halbach foundation, administered by the German Scholars organization.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Alexe, B., Kolaitis, P., Tan, W.-C.: Characterizing schema mappings via data examples. In: Proceedings to the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 261–271 (2010)
3. Arenas, M., Fan, W., Libkin, L.: On the complexity of verifying consistency of XML specifications. SIAM J. Comput. **38**(3), 841–880 (2008)
4. Arenas, M., Libkin, L.: A normal form for XML documents. ACM Trans. Database Syst. **29**(1), 195–232 (2004)
5. Armstrong, W.W.: Dependency structures of database relationships. Inf. Process. **74**, 580–583 (1974)
6. Atzeni, P., Morfuni, N.: Functional dependencies and constraints on null values in database relations. Inf. Control **70**(1), 1–31 (1986)
7. Beeri, C., Bernstein, P.: Computational problems related to the design of normal form relational schemas. ACM Trans. Database Syst. **4**(1), 30–59 (1979)
8. Beeri, C., Bernstein, P.A., Goodman, N.: A sophisticate's introduction to database normalization theory. In: Proceedings of the Fourth International Conference on Very Large Data Bases (VLDB), IEEE Computer Society, pp. 113–124 (1978)
9. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of Armstrong relations for functional dependencies. J. ACM **31**(1), 30–46 (1984)
10. Beeri, C., Mendelzon, A.O., Sagiv, Y., Ullman, J.D.: Equivalence of relational database schemes. SIAM J. Comput. **10**(2), 352–370 (1981)
11. Bernstein, P.: Synthesizing third normal form relations from functional dependencies. ACM Trans. Database Syst. **1**(4), 277–298 (1976)
12. Bernstein, P.A., Goodman, N.: What does Boyce-Codd Normal Form do? In: Proceedings of the Sixth International Conference on Very Large Data Bases (VLDB), pp. 245–259 (1980)
13. Beskales, G., Ilyas, I., Golab, L.: Sampling the repairs of functional dependency violations under hard constraints. Proc. VLDB **3**(1), 197–207 (2010)
14. Bhargava, G., Goel, P., Iyer, B.: Hypergraph-based reorderings of outer join queries with complex predicates. In: Proceedings of the International Conference on Management of Data (SIGMOD), ACM, pp. 304–315 (1995)
15. Bisbal, J., Grimson, J.: Database sampling with functional dependencies. Inf. Softw. Technol. **43**(10), 607–615 (2001)
16. Biskup, J.: Security in computing systems. Springer, Berlin (2009)
17. Biskup, J., Dayal, U., Bernstein, P.: Synthesizing independent database schemas. In: Proceedings of the International Conference on Management of Data (SIGMOD), pp. 143–151 (1979)
18. Biskup, J., Embley, D., Lochner, J.: Reducing inference control to access control for normalized database schemas. Inf. Proc. Lett. **106**(1), 8–12 (2008)
19. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Int. J. Inf. Sec. **7**(3), 199–217 (2008)
20. Boehm, B.: Software Engineering Economics. Prentice Hall, NJ (1981)
21. Bojanczyk, M., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and XML reasoning. J. ACM 56(3) (2009)
22. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.: Reasoning about keys for XML. Inf. Syst. **28**(8), 1037–1063 (2003)
23. CA Technologies: ERwin Data Modeler-methods guide. https://support.ca.com/cadocs/0/e002961e.pdf, p. 86 (2011)
24. Cali, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: Data integration under integrity constraints. Inf. Syst. **29**(2), 147–163 (2004)
25. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), pp. 155–160 (2001)
26. Cheng, Q., Gryz, J., Koo, F., Leung, C., Liu, L., Qian, X., Schiefer, K. Implementation of two semantic query optimization techniques in DB2 universal database. In: Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), pp. 687–698 (1999)
27. Chomicki, J.: Consistent query answering: Five easy pieces. In: Proceedings of the 11th International Conference on Database Theory, no. 4353 in Lecture Notes in Computer Science, Springer, pp. 1–17 (2007)
28. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970)
29. Codd, E.F.: Further normalization of the database relational model. In: Proceedings of the Courant Computer Science Symposia 6: Data Base Systems, pp. 33–64 (1972)
30. Codd, E.F.: Understanding relations. ACM SIGFIDET FDT Bull. **7**(3–4), 23–28 (1975)
31. Codd, E.F.: Extending the database relational model to capture more meaning. ACM Trans. Database Syst. **4**(4), 397–434 (1979)
32. Cohen, E.S.: The theory, practice and methodology of relational database design and programming. http://www.openlineconsult.com/, (2008)
33. Date, C.: Database in depth: relational theory for practitioners, 1 ed. O'Reilly Media, Sebastopol (2005)
34. Davidson, S., Fan, W., Hara, C.: Propagating XML constraints to relations. J. Comput. Syst. Sci. **73**(3), 316–361 (2007)
35. De Marchi, F., Lopes, S., Petit, J.-M., Toumani, F.: Analysis of existing databases at the logical level: the DBA companion project. SIGMOD Rec **32**(1), 47–52 (2003)
36. De Marchi, F., Petit, J.-M.: Semantic sampling of existing databases through informative Armstrong databases. Inf. Syst. **32**(3), 446–457 (2007)
37. Delobel, C., Adiba, M.: Relational database systems. North Holland, (1985)

38. Demetrovics, J.: On the equivalence of candidate keys with Sperner systems. Acta Cybern. **4**, 247–252 (1980)
39. Demetrovics, J., Katona, G., Miklos, D., Thalheim, B.: On the number of independent functional dependencies. In: Proceedings of the 4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS), no. 3861 in Lecture Notes in Computer Science, Springer, pp. 83–91 (2006)
40. Demetrovics, J., Thi, V.: Armstrong relations, functional dependencies and strong dependencies. Comput. Artif. Intell. 14(3) (1995)
41. Demetrovics, J., Thi, V.: Some remarks on generating Armstrong and inferring functional dependencies relation. Acta Cybern. **12**(2), 167–180 (1995)
42. Deutsch, A., Ludäscher, B., Nash, A.: Rewriting queries using views with access patterns under integrity constraints. Theor. Comput. Sci. **371**(3), 200–226 (2007)
43. Deutsch, A., Popa, L., Tannen, V.: Query reformulation with constraints. SIGMOD Rec. **35**(1), 65–73 (2006)
44. Diederich, J., Milton, J.: New methods and fast algorithms for database normalization. ACM Trans. Database Syst. **13**(3), 339–365 (1988)
45. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 6 ed. Addison-Wesley, Reading (2010)
46. Enders, A., Romback, H.: A Handbook of Software and Systems Engineering; Empirical Overservations, Laws and Theories. Addison-Wesley, Reading (2003)
47. Engel, K.: Sperner theory. Cambridge University Press, Cambridge (1997)
48. Fagin, R.: The decomposition versus the synthetic approach to relational database design. In: Proceedings of the Third International Conference on Very Large Data Bases (VLDB), pp. 441–446 (1977)
49. Fagin, R.: Armstrong databases. Technical report RJ3440(40926), IBM Research Laboratory, San Jose, California, USA (1982)
50. Fagin, R.: Horn clauses and database dependencies. J. ACM **29**(4), 952–985 (1982)
51. Fagin, R., Kolaitis, P., Popa, L., Tan, W.: Reverse data exchange: coping with nulls. In: Proceedings of the Twenty-Eight ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 23–32 (2009)
52. Fagin, R., Vardi, M.: Armstrong databases for functional and inclusion dependencies. Inf. Process. Lett. **16**(1), 13–19 (1983)
53. Fagin, R., Vardi, M.: The theory of data dependencies—an overview. In: Proceedings of the 11th Colloqium on Automata, Languages and Programming (ICALP), no. 172 in Lecture Notes in Computer Science, Springer, pp. 1–22 (1984)
54. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst. 33(2) (2008)
55. Fan, W., Geerts, F., Xibei, J.: A revival of integrity constraints for data cleaning. PVLDB 1(2):1522–1523 (2008)
56. Fan, W., Geerts, F., Xibei, J.: Semandaq: a data quality system based on conditional functional dependencies. Proc. VLDB 1(2): 1460–1463 (2008)
57. Fan, W., Ma, S., Hu, Y., Liu, J., Wu, Y.: Propagating functional dependencies with conditions. Proc. VLDB **1**(1), 391–407 (2008)
58. Farkas, C., Jajodia, S.: The inference problem: a survey. SIGKDD Explor. **4**(2), 6–11 (2002)
59. Fischer, P.C., Saxton, L.V., Thomas, S.J., Van Gucht, D.: Interactions between dependencies and nested relational structures. J. Comput. Syst. Sci. **31**(3), 343–354 (1985)
60. Ginsburg, S., Hull, R.: Characterizations for functional dependency and Boyce-Codd normal form families. Theor. Comput. Sci. **26**, 243–286 (1983)
61. Gottlob, G., Libkin, L.: Investigation on Armstrong relations, dependency inference, and excluded functional dependencies. Acta Cybern. **9**(4), 385–402 (1990)
62. Gottlob, G., Pichler, R., Wei, F.: Tractable database design through bounded treewidth. Inf. Syst. **35**(3), 278–298 (2010)
63. Gottlob, G., Zicari, R.: Closed world databases opened through null values. In: Proceedings of the Fourteenth International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, pp. 50–61 (1988)
64. Grahne, G.: Dependency satisfaction in databases with incomplete information. In: Proceedings of the Tenth International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, pp. 37–45 (1984)
65. Grant, J.: Null values in a relational data base. Inf. Process. Lett. **6**(5), 156–157 (1977)
66. Hartmann, S., Link, S.: Characterising nested database dependencies by fragments of propositional logic. Ann. Pure Appl. Logic **152**(1–3), 84–106 (2008)
67. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
68. Hartmann, S., Link, S.: Numerical constraints on XML data. Inf. Comput. **208**(5), 521–544 (2010)
69. Hartmann, S., Link, S.: When data dependencies over SQL tables meet the Logics of Paradox and S-3. In: Proceedings to the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 317–326 (2010)
70. Hasan, W., Pirahesh, H.: Query rewrite optimization in STARBURST. Technical Report RJ6367, IBM Research Laboratory, San Jose, California, USA (1988)
71. Ilyas, I., Markl, V., Haas, P., Brown, P., Aboulnaga, A.: CORDS: Automatic discovery of correlations and soft functional dependencies. In: Proceedings of the International Conference on Management of Data (SIGMOD), ACM, pp. 647–658 (2004)
72. Imielinski, T., Lipski, W. Jr..: Incomplete information in relational databases. J. ACM **31**(4), 761–791 (1984)
73. Imielinski, T., Van der Meyden, R., Vadaparty, K.: Complexity tailored design: a new design methodology for databases with incomplete information. J. Comput. Syst. Sci. **51**(3), 405–432 (1995)
74. Jensen, C., Snodgrass, R., Soo, M.: Extending existing dependency theory to temporal databases. IEEE Trans. Knowl. Data Eng. **8**(4), 563–582 (1996)
75. Katona, G., Tichler, K.: Some contributions to the minimum representation problem of key systems. In: Proceedings of the 4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS), vol. 3861 of Lecture Notes in Computer Science, Springer, pp. 240–257 (2006)
76. Kim, M.-S., Whang, K.-Y., Lee, J.-G., Lee, M.-J.: Structural optimization of a full-text *n*-gram index using relational normalization. VLDB J. **17**(6), 1485–1507 (2008)
77. Kimura, H., Huo, G., Rasin, A., Madden, S., Zdonik, S.: Correlation maps: a compressed access method for exploiting soft functional dependencies. Proc. VLDB **2**(1), 1222–1233 (2009)
78. Klug, A., Price, R.: Determining view dependencies using tableaux. ACM Trans. Database Syst. **7**(3), 361–380 (1982)
79. Koch, C., Olteanu, D.: Conditioning probabilistic databases. Proc. VLDB **1**(1), 313–325 (2008)
80. Kolahi, S.: Dependency-preserving normalization of relational and XML data. J. Comput. Syst. Sci. **73**(4), 636–647 (2007)
81. Kolahi, S., Libkin, L.: An information-theoretic analysis of worst-case redundancy in database design. ACM Trans. Database Syst. 35(1) (2010)
82. Langeveldt, W.-D., Link, S.: Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies. Inf. Syst. **35**(3), 352–374 (2010)
83. Lechtenbörger, J., Vossen, G.: Multidimensional normal forms for data warehouse design. Inf. Syst. **28**(5), 415–434 (2003)

84. Levene, M., Loizou, G.: Null inclusion dependencies in relational databases. Inf. Comput. **136**(2), 67–108 (1997)
85. Levene, M., Loizou, G.: Axiomatisation of functional dependencies in incomplete relations. Theor. Comput. Sci. **206**(1-2), 283–300 (1998)
86. Levene, M., Loizou, G.: Database design for incomplete relations. ACM Trans. Database Syst. **24**(1), 80–125 (1999)
87. Libkin, L.: Data exchange and incomplete information. In: Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pp. 60–69 (2006)
88. Libkin, L., Wong, L.: Semantic representations and query languages for Or-sets. J. Comput. Syst. Sciences **52**(1), 125–142 (1996)
89. Lien, E.: On the equivalence of database models. J. ACM **29**(2), 333–362 (1982)
90. Ling, T.: A normal form for Entity-Relationship diagrams. In: Proceedings of the Fourth International Conference on the Entity-Relationship Approach (ER), pp. 24–35 (1985)
91. Link, S.: On the implication of multivalued dependencies in partial database relations. Int. J. Found. Comput. Sci. **19**(3), 691–715 (2008)
92. Lucchesi, C., Osborn, S.: Candidate keys for relations. J. Comput. Syst. Sci. **17**(2), 270–279 (1978)
93. Maier, D.: Minimum covers in relational database model. J. ACM **27**(4), 664–674 (1980)
94. Maier, D.: The Theory of Relational Databases. Computer Science Press, (1983)
95. Makinouchi, A.: A consideration on normal form of not-necessarily-normalised relation in the relational data model. In: Proceedings of the Third International Conference on Very Large Data Bases (VLDB), pp. 447–453 (1977)
96. Mannila, H., Räihä, K.-J.: Design by example: an application of Armstrong relations. J. Comput. Syst. Sci. **33**(2), 126–141 (1986)
97. Mannila, H., Räihä, K.-J.: Design of Relational Databases. Addison-Wesley, Reading (1992)
98. Mannila, H., Räihä, K.-J.: Algorithms for inferring functional dependencies from relations. Data Knowl. Eng. **12**(1), 83–99 (1994)
99. Marnette, B., Mecca, G., Papotti, P.: Scalable data exchange with functional dependencies. Proc. VLDB **3**(1), 105–116 (2010)
100. Martin, J.: Information Engineering. Prentice Hall, NJ (1989)
101. Miklau, G., Suciu, D.: A formal analysis of information disclosure in data exchange. J. Comput. Syst. Sci. **73**(3), 507–534 (2007)
102. Miller, R., Hernandez, M., Haas, L., Yan, L.-L., Ho, C., Fagin, R., Popa, L.: The Clio project: managing heterogeneity. SIGMOD Rec. **30**(1), 78–83 (2001)
103. Molinaro, C., Greco, S.: Polynomial time queries over inconsistent databases with functional dependencies and foreign keys. Data Knowl. Eng. **69**(7), 709–722 (2010)
104. Paulley, G.: Exploiting functional dependence in query optimization. Technical Report UW-CS-2000-11, University of Waterloo, Waterloo, Canada, (2000)
105. Paulley, G., Larson, P.-A.: Exploiting uniqueness in query optimization. In: Proceedings of the Tenth International Conference on Data Engineering (ICDE), IEEE Computer Society, pp. 68–79 (1994)
106. Qian, L., LeFevre, K., Jagadish, H.: CRIUS: user-friendly database design. Proc. VLDB Endow. **4**(2), 81–92 (2011)
107. Rauh, O., Stickel, E.: Standard transformations for the normalization of ER schemata. Inf. Syst. **21**(2), 187–208 (1996)
108. Rosenthal, A., Reiner, D.: Tools and transformations—rigorous and otherwise—for practical database design. ACM Trans. Database Syst. **19**(2), 167–211 (1994)
109. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artif. Intell. **74**, 249–310 (1995)
110. Silva, A., Melkanoff, M.: A method for helping discover the dependencies of a relation. In: Advances in Data Base Theory, pp. 115–133 (1979)
111. Simmen, D., Shekita, E., Malkemus, T.: Fundamental techniques for order optimization. In: Proceedings of the International Conference on Management of Data (SIGMOD), pp. 57–67 (1996)
112. Sözat, M., Yazici, A.: A complete axiomatization for fuzzy functional and multivalued dependencies in fuzzy database relations. ACM Fuzzy Sets Syst. **117**(2), 161–181 (2001)
113. Standish Group: Unfinished voyages. The Standish Group International, available on-line at http://www.standishgroup.com/sample_research/unfinsihed_voyages_1.php (1995)
114. Tan, H., Zhao, Y.: Automated elicitation of functional dependencies from source codes of database transactions. Inf. Softw. Technol. **46**(2), 109–117 (2004)
115. Tari, Z., Stokes, J., Spaccapietra, S.: Object normal forms and dependency constraints for object-oriented schemata. ACM Trans. Database Syst. **22**, 513–569 (1997)
116. Thalheim, B.: Dependencies in relational databases. Teubner, Oxford (1991)
117. Thalheim, B.: Entity–Relationship Modeling. Springer, Heidelberg (2000)
118. Toman, D., Weddell, G.: On keys and functional dependencies as first-class citizens in description logics. J. Autom. Reason. **40** (2–3), 117–132 (2008)
119. Vadaparty, K., Naqvi, S.: Using constraints for efficient query processing in nondeterministic databases. IEEE Trans. Knowl. Data Eng. **7**(6), 850–864 (1995)
120. Vincent, M.: Semantic foundations of 4NF in relational database design. Acta Inf. **36**(3), 173–213 (1999)
121. Vincent, M., Liu, J., Liu, C.: Strong functional dependencies and their application to normal forms in XML. ACM Trans. Database Syst. **29**(3), 445–462 (2004)
122. Wang, D., Dong, X., Sarma, A., Franklin, M., Halevy, A.: Functional dependency generation and applications in pay-as-you-go data integration systems. In: Proceedings of the 12th International Workshop on the Web and Databases (WebDB), ACM (2009)
123. Wang, H., Liu, R.: Privacy-preserving publishing data with full functional dependencies. In: Proceedings of the 15th International Conference on Database Systems for Advanced Applications, Part II (DASFAA), vol. 5982 of Lecture Notes in Computer Science, Springer, pp. 176–183 (2010)
124. Weddell, G.: Reasoning about functional dependencies generalized for semantic data models. ACM Trans. Database Syst. **17**(1), 32–64 (1992)
125. Wijsen, J.: Temporal FDs on complex objects. ACM Trans. Database Syst. **24**(1), 127–176 (1999)
126. Wolf, G., Kalavagattu, A., Khatri, H., Balakrishnan, R., Chokshi, B., Fan, J., Chen, Y., Kambhampati, S.: Query processing over incomplete autonomous databases: query rewriting using learned data dependencies. VLDB J. **18**(5), 1167–1190 (2009)
127. Yu, C., Jagadish, H.: XML schema refinement through redundancy detection and normalization. VLDB J. **17**(2), 203–223 (2008)
128. Zaniolo, C.: Database relations with null values. J. Comput. Syst. Sci. **28**(1), 142–166 (1984)
129. Zultner, R.: The deming way: total quality management for software. In: Proceedings of Total Quality Management for Software, pp. 134–145 (1992)

11 Proofs

Theorem 15 (Theorem 2 restated) *Let Σ be a set of FDs and let R_s be an NFS over R. Then we have:*

1. $\Sigma \models_{R_s}$ *unique*(X) *if and only if* $\Sigma \models_{R_s} X \to R$,
2. $\Sigma \models_{R_s}$ *Codd*(X) *if and only if* $\Sigma \models_{R_s} X \to R$ *and* $X \subseteq R_s$.

Proof The first equivalence is a straight consequence of the fact that a relation satisfies *unique*(X) if and only if it satisfies the FD $X \to R$. For the second equivalence we first show that if $\Sigma \models_{R_s} X \to R$ and $X \subseteq R_s$ hold, then $\Sigma \models_{R_s}$ *Codd*(X) holds as well. In fact, let r denote an arbitrary relation over R that satisfies Σ and R_s. It follows that r satisfies $X \to R$ as well since $\Sigma \models_{R_s} X \to R$. Since $X \subseteq R_s$ holds, we know that r satisfies the NFS X. Consequently, r satisfies *Codd*(X).

It remains to show that if $\Sigma \not\models_{R_s} X \to R$ or $X \not\subseteq R_s$, then $\Sigma \not\models_{R_s}$ *Codd*(X).

Suppose first that $\Sigma \not\models_{R_s} X \to R$. Then there is some relation r over R that satisfies Σ and the NFS R_s, but violates the FD $X \to R$. Consequently, there are two tuples $t, t' \in r$ such that $t[X] = t'[X]$, t and t' are X-total and $t \neq t'$. The two-tuple relation $r' = \{t, t'\}$ shows that $\Sigma \not\models_{R_s}$ *Codd*(X) since r' satisfies Σ and R_s (since $r' \subseteq r$ holds), but violates *Codd*(X).

Suppose now that $X \not\subseteq R_s$ holds, i.e., $X - R_s$ is non-empty. In this case we define a single-tuple relation $r := \{t\}$ for some tuple t over R such that $t[A] :=$ `ni` for all $A \in R - R_s$ and $t[B] \in dom(B) - \{\texttt{ni}\}$ for all $B \in R_s$. It follows that r satisfies Σ and the NFS R_s, but r violates *Codd*(X) since $X \cap (R - R_s) \neq \emptyset$. □

Theorem 16 (Theorem 4 restated) *Let R be some relation schema, let Σ be a set of standard FDs and let R_s be an NFS over R. For all relations r over R it holds that r is an Armstrong table for Σ and R_s if and only if both of the following conditions are satisfied:*

1. *for all non-empty $X \subseteq R$ we have*
$$X^*_{\Sigma,R_s} = \bigcap\{w(Z) \mid X \subseteq Z \in ag^s(r)\}, \textit{and}$$
2. *total*$(r) = R_s$.

Proof **Sufficiency.** Let r be a relation over R that satisfies the conditions. We show that r is an Armstrong table for Σ and R_s.

Let $X \to A \in \Sigma$. That is, $A \in X^*_{\Sigma,R_s}$. Assume that there are distinct tuples $t, t' \in r$ such that $t[X] = t'[X]$ and t, t' are X-total. That is, $X \subseteq X' = ag^s(t, t')$. Hence, the first condition shows that $A \in w(X')$, and thus $A \in ag^w(t, t')$. Therefore, $t[A] = t'[A]$ holds. We have shown that r satisfies Σ.

Let $X \to A \notin \Sigma^*_{R_s}$. Hence, $A \notin X^*_{\Sigma,R_s}$. By the first condition there is some $Z \in ag^s(r)$ such that $X \subseteq Z$ and $A \notin w(Z)$. In particular, there are tuples t, t' such that $X \subseteq Z = ag^s(t, t')$ and $A \notin ag^w(t, t')$. That is, we have $t[X] = t'[X]$, t, t' are X-total and $t[A] \neq t'[A]$. This shows that r violates every FD not in $\Sigma^*_{R_s}$.

Finally, the condition *total*$(r) = R_s$ ensures that r satisfies every NFS implied by R_s and violates every NFS not implied by R_s. Consequently, r is indeed an Armstrong table for Σ and R_s.

Necessity. Let r be a relation over R that is Armstrong for Σ and R_s. We show that r satisfies the conditions. Let $t, t' \in r$ be distinct tuples such that $X \subseteq X' = ag^s(t, t')$. As r satisfies $\Sigma^*_{R_s}$ we have $X^*_{\Sigma,R_s} \subseteq ag^w(t, t')$, and thus $X^*_{\Sigma,R_s} \subseteq w(X')$. Therefore, $X^*_{\Sigma,R_s} \subseteq \bigcap\{w(Z) \mid X \subseteq Z \in ag^s(r)\}$ holds.

Next we show that $X^*_{\Sigma,R_s} \supseteq \bigcap\{w(Z) \mid X \subseteq Z \in ag^s(r)\}$. Assume there is an $A \notin X^*_{\Sigma,R_s}$ such that $A \in \{w(Z) \mid X \subseteq Z \in ag^s(r)\}$. Then we have $A \in ag^w(t, t')$ for all distinct tuples $t, t' \in r$ with $X \subseteq Z = ag^s(t, t')$. That is, r satisfies $X \to A$. This, however, contradicts the assumption $A \notin X^*_{\Sigma,R_s}$ since r is Armstrong for Σ and R_s. Consequently, $X^*_{\Sigma,R_s} \supseteq \bigcap\{w(Z) \mid X \subseteq Z \in ag^s(r)\}$ holds. Finally, since r is Armstrong for Σ and R_s it follows that *total*$(r) = R_s$. □

Theorem 17 (Theorem 5 restated) *Let R be some relation schema, let Σ be a set of standard FDs and let R_s be an NFS over R. For all relations r over R it holds that r is an Armstrong table for Σ and R_s if and only if the following conditions are satisfied:*

1. $\forall A \in R \forall X \in max_{\Sigma,R_s}(A)(X \in ag^s(r) \wedge A \notin w(X))$,
2. $\forall X \in ag^s(r)(X^*_{\Sigma,R_s} \subseteq w(X))$, *and*
3. *total*$(r) = R_s$.

Proof **Sufficiency.** Let r be some relation over R that satisfies conditions 1., 2. and 3. We show that r is an Armstrong table for Σ and R_s.

Let $X \to A \in \Sigma$. Assume that there are distinct $t, t' \in r$ such that $t[X] = t'[X]$ and t is X-total. That is, $X \subseteq X' = ag^s(t, t')$. Note that $A \in (X')^*_{\Sigma,R_s}$ by soundness of the augmentation rule. Hence, condition 2. implies that $A \in w(X')$. In particular, $A \in ag^w(t, t')$. Therefore, $t[A] = t'[A]$. Hence, r satisfies Σ.

Let $X \to A \notin \Sigma^*_{R_s}$. It follows that there is some $X' \in max_{\Sigma,R_s}(A)$ such that $X \subseteq X'$ and $A \notin (X')^*_{\Sigma,R_s}$. Condition 1. implies that $X' \in ag^s(r)$ and $A \notin w(X')$. Hence, there is some $Y \in ag^w(r)$ such that $(X', Y) \in ag(r)$ and $A \notin Y$. This shows that there are two distinct $t, t' \in r$ such that $t[X'] = t'[X']$ and t, t' are X'-total and $t[A] \neq t'[A]$. We have shown that r violates every functional dependency that is not in $\Sigma^*_{R_s}$.

Condition 3. ensures that r satisfies every NFS implied by R_s, and violates every NFS not implied by R_s. Consequently, r is an Armstrong table for Σ and R_s.

Necessity. Let r be some relation over R that is an Armstrong table for Σ and R_s. We show that r satisfies conditions 1., 2. and 3.

Let $A \in R$, and let $X \in max_{\Sigma,R_s}(A)$. That is, $\Sigma \not\models_{R_s} X \to A$ and for all $B \in R - X$ it is true that $\Sigma \models_{R_s} XB \to A$. Since r is an Armstrong table for Σ and R_s it follows that r violates $X \to A$ and for all $B \in R - X$ that r satisfies the FD $XB \to A$. The violation of $X \to A$ implies that there are distinct $t, t' \in r$ such that $X \subseteq ag^s(t,t')$ and $A \notin ag^w(t,t')$. If there was some attribute C of R in $ag^s(t,t') - X$, then r would violate the FD $XC \to A$. Consequently, $X = ag^s(t,t')$. We have just shown that for every $A \in R$ and for every $X \in max_{\Sigma,R_s}(A)$ it is true that $X \in ag^s(r)$ and $A \notin w(X)$, i.e., condition 1. holds.

Next we show that r satisfies condition 2. Therefore, let $X \in ag^s(r)$. We need to show that $X^*_{\Sigma,R_s} \subseteq w(X)$. Let A be some attribute of R such that $A \notin w(X)$. That is, there is some $Y \in ag^w(r)$ such that $(X,Y) \in ag(r)$ and $A \notin Y$. Consequently, there are some distinct $t, t' \in r$ such that $X = ag^s(t,t')$ and $A \notin ag^w(t,t')$. That is, r violates the FD $X \to A$. Since r is an Armstrong table for Σ and R_s it follows that $A \notin X^*_{\Sigma,R_s}$. We have just shown that $X^*_{\Sigma,R_s} \subseteq w(X)$.

Since r is an Armstrong table for Σ and R_s it follows that $total(r) = R_s$. $\square$

Theorem 18 (Theorem 6 restated) *Let R be some relation schema, let Σ be an FD set, and R_s an NFS over R. For all relations r over R it holds that r satisfies Σ and R_s if and only if $R_s \subseteq total(r)$ and $\forall X \in ag^s(r)(X^*_{\Sigma,R_s} \subseteq w(X))$ holds.*

Proof **Sufficiency.** The proof of Theorem 5 shows that r satisfies Σ, if $\forall X \in ag^s(r)(X^*_{\Sigma} \subseteq w(X))$ holds. Furthermore, if $R_s \subseteq total(r)$, then r is R_s-total.

Necessity. If r satisfies R_s, then $R_s \subseteq total(r)$. It remains to show that $\forall X \in ag^s(r)(X^*_{\Sigma,R_s} \subseteq w(X))$ holds, if r satisfies Σ and R_s. Assume that there is some $X \in ag^s(r)$ and there is some $A \in R$ such that $A \in X^*_{\Sigma,R_s} - w(X)$. Since $A \notin w(X)$, there is some $Y \subseteq R$ such that $(X,Y) \in ag(r)$ and $A \notin Y$. Consequently, there are distinct $t, t' \in r$ such that $X = ag^s(t,t')$ and $Y = ag^w(t,t')$. That is, $t[X] = t'[X]$ and t, t' are X-total, and $t[A] \neq t'[A]$. Hence, r violates the FD $X \to A$. From $A \in X^*_{\Sigma,R_s}$ it follows that $X \to A \in \Sigma^*_{R_s}$. The definition of implication shows that r violates Σ. $\square$

Theorem 19 (Theorem 7 restated) *Let R be a relation schema, R_s a null-free subschema over R, and $\Sigma = \Sigma' \cup \{X \to A\}$ a set of standard functional dependencies over R. For $C \in R$ let $V \in max_{\Sigma,R_s}(C)$. Then $V \in max_{\Sigma',R_s}(C)$ or ($C = A$ or $A \in R_s$) holds and there is some $B \in X - V$ such that*

i) $VB \in max_{\Sigma',R_s}(C)$, if $X \not\subseteq R_s$, or
ii) $V = W \cap Z$ for some $W \in max_{\Sigma',R_s}(C)$ and some $Z \in max_{\Sigma',R_s}(B)$.

The proof uses the following simple observation:

Remark 4 Let $\Sigma = \Sigma' \cup \{X \to A\}$ and $U \subseteq R$. When $A \in U^*_{\Sigma,R_s}$ we have

$$U^*_{\Sigma,R_s} = \begin{cases} (UA)^*_{\Sigma,R_s} = (UA)^*_{\Sigma',R_s} & \text{, if } A \in R_s, \\ U^*_{\Sigma',R_s}A & \text{, otherwise,} \end{cases}$$

while $U^*_{\Sigma,R_s} = U^*_{\Sigma',R_s}$ holds when $A \notin U^*_{\Sigma,R_s}$. Furthermore, the following statements are equivalent: a) $U^*_{\Sigma',R_s} \subset U^*_{\Sigma,R_s}$, b) $A \in U^*_{\Sigma,R_s} - U^*_{\Sigma',R_s}$, and c) $A \notin U^*_{\Sigma',R_s}$ and $X \subseteq UR_s \cap U^*_{\Sigma',R_s}$, cf. Algorithm 1. $\square$

Proof From $V \in max_{\Sigma,R_s}(C)$ and $V \subseteq V^*_{\Sigma',R_s} \subseteq V^*_{\Sigma,R_s}$ we get $C \notin V^*_{\Sigma,R_s}$ and $C \notin V^*_{\Sigma',R_s}$. If $V \in max_{\Sigma',R_s}(C)$ we are done. Otherwise, there is some $W \in max_{\Sigma',R_s}(C)$ with $V \subset W$. By $V \in max_{\Sigma,R_s}(C)$ we obtain $C \in W^*_{\Sigma,R_s} - W^*_{\Sigma',R_s}$, so that $W^*_{\Sigma',R_s} \subset W^*_{\Sigma,R_s}$. When applying Remark 4 to the set W, we note $A \in W^*_{\Sigma,R_s} - W^*_{\Sigma',R_s}$ and, therefore, ($C = A$ or $A \in R_s$).

Remark 4 for W further yields $A \notin W^*_{\Sigma',R_s}$ and $X \subseteq WR_s \cap W^*_{\Sigma',R_s}$. Hence, $A \notin V^*_{\Sigma',R_s}$ as $V \subset W$. Assume $A \in V^*_{\Sigma,R_s} - V^*_{\Sigma',R_s}$. Then, $C \neq A$ as $C \notin V^*_{\Sigma,R_s}$, and thus $A \in R_s$. When applying Remark 4 to the set V, we observe $V^*_{\Sigma,R_s} = (VA)^*_{\Sigma,R_s} = (VA)^*_{\Sigma',R_s}$. Hence, $C \notin (VA)^*_{\Sigma,R_s}$. By $V \in max_{\Sigma,R_s}(C)$ we obtain $A \in V$. Thus $A \in V \subseteq W \subseteq W^*_{\Sigma',R_s}$ which contradicts $A \in W^*_{\Sigma,R_s} - W^*_{\Sigma',R_s}$. Hence, our assumption is false, and $A \notin V^*_{\Sigma,R_s}$ holds. From $A \notin V^*_{\Sigma,R_s}$ and Remark 4 for V we conclude $V^*_{\Sigma,R_s} = V^*_{\Sigma',R_s}$, and $X \not\subseteq VR_s \cap V^*_{\Sigma',R_s}$, that is, ($X \not\subseteq VR_s$ or $X \not\subseteq V^*_{\Sigma',R_s}$). By $X \subseteq WR_s$ we obtain $((X - R_s) \cap (W - V) \neq \emptyset$ or $X \not\subseteq V^*_{\Sigma',R_s})$. Therefore, V is a subset of a member U of

$$\begin{aligned}\mathcal{V} := &\{W - B : B \in X - R_s\} \cup \\ &\{W \cap Z : W \not\subseteq Z \in \textstyle\bigcup_{B \in X} max_{\Sigma',R_s}(B)\}.\end{aligned}$$

From $U \subset W$ and $W \in max_{\Sigma',R_s}(C)$ we get $C \notin U^*_{\Sigma',R_s}$. By definition of $\mathcal{V}$, we observe $X \not\subseteq UR_s \cap U^*_{\Sigma',R_s}$ which yields $U^*_{\Sigma,R_s} = U^*_{\Sigma',R_s}$ by Remark 4. Thus, we obtain $C \notin U^*_{\Sigma,R_s}$. From $V \in max_{\Sigma,R_s}(C)$ we derive $V = U$, that is, V itself is a member of $\mathcal{V}$. This concludes the proof of Theorem 7. $\square$

Theorem 20 (Theorem 11 restated) *Algorithm 10, on input (R, Σ, R_s), computes an Armstrong table for Σ and R_s.*

Proof Let r denote the output of Algorithm 10. We show first that the output r of Algorithm 10 is a subsumption-free relation. Let $t, t' \in r$ denote two distinct tuples. Suppose t results from some $X \in max(A)$ and t' results from some $Y \in max(A)$. The construction guarantees that $t[A] \neq t'[A]$ and $t[A] \neq \texttt{ni} \neq t'[A]$. Hence, neither of t, t' subsumes the other. Suppose that t results from some $X \in max(A)$ and t' results from some $Y \in max(B)$ where $A \neq B$. If $t'[A] \neq \texttt{ni}$, then $t[A] \neq t'[A]$ and $t[A] \neq \texttt{ni} \neq t'[A]$. If $t[B] \neq \texttt{ni}$, then $t[B] \neq t'[B]$ and $t[B] \neq \texttt{ni} \neq t'[B]$. If $t'[A] = \texttt{ni}$ and $t[B] = \texttt{ni}$, then neither of t, t' subsumes the other. Finally, if t results from some $X \in max(A)$ and t' results from step $(A7)$, or if t and t' result from step $(A7)$, then the construction (distinct values) guarantees that neither of t, t' subsumes the other. Hence, r is a subsumption-free relation.

We show r to be an Armstrong table for Σ and R_s.

Let $X \to A \in \Sigma$. Assume that r violates $X \to A$. Then there are distinct $t, t' \in r$ such that $t[X] = t'[X]$ and t, t' are X-total and $t[A] \neq t'[A]$. Since $X \neq \emptyset$, it follows from construction that $\{t, t'\} = \{t_{2i-1}, t_{2i}\}$ for some positive integer i. According to the steps $(A3)$ and $(A4)$ we conclude that there is some $X' \in max(R)$ and there is some $B \in Z := \{C \in R \mid X' \in max(C)\}$ such that $X \subseteq X'$ and $A \in ZR_s$. Suppose that $A \in Z$. Consequently, $X \subseteq X' \in max(A)$, i.e., $A \notin (X')^*_{\Sigma,R_s}$. However, due to the soundness of the augmentation rule we conclude that $A \notin X^*_{\Sigma,R_s}$. This means that $X \to A \notin \Sigma^*_{R_s}$ which contradicts $X \to A \in \Sigma$. Suppose now that $A \in R_s - Z$. From $X' \in max(B)$ it follows that $X' \to B \notin \Sigma^*_{R_s}$ and that $X'A \to B \in \Sigma^*_{R_s}$. From $X \to A \in \Sigma^*_{R_s}$ follows $X' \to A \in \Sigma^*_{R_s}$ by the soundness of the augmentation rule. From the soundness of the reflexivity axiom and the union rule we conclude that $X' \to X'A \in \Sigma^*_{R_s}$. An application of the null transitivity rule to $X' \to X'A$, $X'A \to B$ and $A \in R_s$ results in $X' \to B$. Due to the soundness of the null transitivity rule we conclude that $X' \to B \in \Sigma^*_{R_s}$. This contradicts the fact that $X' \in max(B)$. We have just shown that r satisfies Σ. The construction in steps $(A3)$ and $(A4)$ ensures that r satisfies the NFS R_s.

It is not difficult to see that the relation r violates all standard FDs $X \to A \notin \Sigma^*_{R_s}$. In fact, by definition of $max(A)$ there is some $X' \subseteq R$ such that $X' \in max(A)$ and $X \subseteq X'$. Step $(A4)$ guarantees that there are some distinct $t, t' \in r$ such that $t[X'] = t'[X']$, t, t' are X'-total and $t[A] \neq t'[A]$. Hence, r violates $X \to A$.

It is now quite easy to see that the relation r is an Armstrong table for Σ and R_s. In fact, step $(A7)$ guarantees that r is total on precisely those attributes of R that belong to R_s. Note that $R_s \subseteq total(r)$ always holds due to the construction. Hence, if $total(r) - R_s \neq \emptyset$, then we need to add some tuples with occurrences of $\texttt{ni}$ in all columns $A \in total(r) - R_s$. If $R_s = \emptyset$, $total(r) = R$ and $|R| > 1$, then we require two tuples to ensure that r remains subsumption-free. Otherwise, we can just add a single tuple with occurrences of $\texttt{ni}$ in all columns $A \in total(r) - R_s$. Hence, r satisfies precisely those null-free subschema constraints implied by R_s (namely the subsets of R_s). □

Proposition 5 (Proposition 1 restated) *Let Σ be a set of standard FDs, let R_s be some NFS over some relation schema R, and let r be an Armstrong table for Σ and R_s. Then $|max_{\Sigma,R_s}(R)| \leq |ag(r)| \leq \binom{|r|}{2}$.*

Proof The first condition of Theorem 5 implies that $|max_{\Sigma,R_s}(R)| \leq |ag^s(r)|$. Moreover, $|ag^s(r)| \leq |ag(r)|$, and $|ag(r)| \leq \binom{|r|}{2}$ since every distinct pair of distinct tuples in r has precisely one agree set. □

Proposition 6 (Proposition 2 restated) *The complexity of finding an Armstrong table, given a set of standard FDs and an NFS, is precisely exponential in the number of attributes.*

Proof The time complexity of Algorithm 10 is dominated by that of Algorithm 8 which runs clearly in time exponential in the number of attributes.

It remains to show that there is a set Σ of standard FDs and an NFS R_s for which the number of tuples in each Armstrong table for Σ and R_s is exponential in the number of attributes. According to Proposition 1 it suffices to find a set Σ of standard FDs such that $max_{\Sigma,R_s}(R)$ is exponential in the number of attributes. Such a set Σ is given by

$$\bigcup_{1 \leq i \leq n} \{\{A_{2i-1}, A_{2i}\} \to B\}$$

and the NFS $R_s = A_1 \cdots A_{2n}B$. This is the same set that Beeri, Dowd, Fagin and Statman used to show that the time complexity of finding an Armstrong relation for FDs over total relations takes at least exponential time in the number of attributes [9]. This set works here for the same purpose since all FDs in Σ have the same right-hand side. □

Proposition 7 (Proposition 3 restated) *Let Σ be a set of standard FDs, let R_s be some NFS over some relation schema R, and let r be a minimum-sized Armstrong table for Σ and R_s. Then*

$$\frac{\sqrt{1 + 8 \cdot |max_{\Sigma,R_s}(R)|}}{2} \leq |r| \leq 2 \times |max_{\Sigma,R_s}(R)| + 1.$$

Proof The lower bound follows from Proposition 1. Indeed, it follows that $|max_{\Sigma,R_s}(R)| \leq \binom{|r|}{2}$. Consequently, we have that $\frac{\sqrt{1+8\cdot|max_{\Sigma,R_s}(R)|}}{2} \leq |r|$. The upper bound $2\times|max_{\Sigma,R_s}(R)|+2$ follows immediately from Theorem 11. However, Algorithm 10 outputs an Armstrong table of size $2\times|max_{\Sigma,R_s}(R)|+2$ if and only if $total(r)=R$ holds before step (A7) and $R_s=\emptyset$. We have $B\in total(r)$ before step (A7) if and only if $B\in XA\cup R_s$ for every maximal set $X\in max_{\Sigma,R_s}(A)$ and all $A\in R$. Therefore, we have $total(r)=R$ before step (A7) and $R_s=\emptyset$ if and only if $max_{\Sigma,R_s}(A)=\{R-A\}$ for all $A\in R$ and $R_s=\emptyset$. This again holds if and only if $\Sigma=\emptyset$ and $R_s=\emptyset$. In this special case and when $|R|>1$, one may use the Algorithm from Mannila and Räihä [96] to compute an Armstrong relation for Σ of size $max_{\Sigma,R_s}(R)+1=|R|+1$ and add two tuples according to step (A7) of Algorithm 10 to obtain an Armstrong table for Σ and R_s. If $|R|=1$, then the singleton relation consisting of the tuple $t=$ `ni` is an Armstrong table for Σ and R_s. □

Theorem 21 (Theorem 12 restated) *There is some relation schema R, some NFS R_s and some standard FD set Σ over R such that Algorithm 10 computes a minimum-sized Armstrong table for Σ and R_s.*

Proof Let $R=A_1\cdots A_n$ with $n\geq 3$, $R_s=\emptyset$, and Σ consist of the FDs $A_i\to A_{i+1}$ with $i=1,\ldots,n$. We find $max_{\Sigma,R_s}(A_i)=\{R-A_{i-1}A_i\}$ for $i=1,\ldots,n$ (with the convention that $A_i=A_{i+n}=A_{i-n}$). Let r be any Armstrong table for Σ and R_s. We show that its size it at least $2\times|max_{\Sigma,R_s}(R)|$. By Theorem 5 there are tuples $t_i,t'_i\in r$ for all $i=1,\ldots,n$ such that $ag^s(t_i,t'_i)=R-A_{i-1}A_i$ and $t_i[A_i]\neq t'_i[A_i]$. We conclude $t_i[A_{i-1}]=ni=t'_i[A_{i-1}]$ for $i=1,\ldots,n$ since r satisfies the FD $A_{i-2}\to A_{i-1}$, and also $t_i[A_i]\neq ni\neq t'_i[A_i]$ for $i=1,\ldots,n$ since r is subsumption-free. Hence, $total(\{t_i\})=R-A_{i-1}=total(\{t'_i\})=R-A_{i-1}$ for $i=1,\ldots,n$. As $t_1,t'_1,\ldots,t_n,t'_n$ are mutually distinct, r has size at least $2\times n=2\times|max_{\Sigma,R_s}(R)|$. □

Theorem 22 (Theorem 13 restated) *There is some relation schema R, some NFS R_s and some standard FD set Σ over R such that there is an Armstrong table for Σ and R_s whose size is in $\mathcal{O}(n)$, and the optimal cover of Σ with respect to R_s has size $\mathcal{O}(2^n)$.*

Proof Let $R=A_1B_1\cdots A_nB_nC$, $R_s=R$ and

$$\Sigma=\{X_1,\ldots,X_n\to C\mid\forall i=1,\ldots,n(X_i\in\{A_i,B_i\})\}.$$

We show that Σ is the optimal cover of Σ with respect to R_s, but there is an Armstrong table for Σ and R_s where the number of tuples is in $\mathcal{O}(n)$.

We will show first that Σ is non-redundant (no subset of Σ implies all FDs in Σ), and then show that Σ is an optimal cover of itself. We note that for every FD $\sigma\in\Sigma$, where $X=LHS(\sigma)$ denotes the attribute set on the left-hand side of σ, the closure $X^*_{(\Sigma-\{\sigma\},R_s)}$ of X with respect to $\Sigma-\{\sigma\}$ and R_s is X itself, i.e., $X^*_{(\Sigma-\{\sigma\},R_s)}=X$. The reason is that there is no $\sigma'\in\Sigma-\{\sigma\}$ such that $LHS(\sigma')\subseteq X$. Hence, $C\notin X^*_{(\Sigma-\{\sigma\},R_s)}$ and we conclude that σ is not implied by $\Sigma-\{\sigma\}$ and R_s. That is, Σ is non-redundant. Next we remark that every optimal cover Σ' of Σ with respect to R_s contains only FDs $X\to Y$ such that $Y=C$. Suppose, to the contrary, that there is some FD $X\to Y$ in Σ' such that $Y\neq C$. If $Y-X=C$ and $Y\cap X\neq\emptyset$, then Σ' is not optimal since $(\Sigma'-\{X\to Y\})\cup\{X\to Y-X\}$ is equivalent to Σ but contains less attributes than Σ'. If $Y-X=\emptyset$, then $\Sigma'-\{X\to Y\}$ is equivalent to Σ but contains less symbol occurrences than Σ'. If $Y-X\neq\emptyset$ and $Y-X\neq C$, then $\Sigma\not\models_{R_s}X\to Y$ and, therefore, Σ' is not a cover of Σ with respect to R_s. Moreover, every FD $X\to Y$ in an optimal cover Σ' of Σ with respect to R_s satisfies that $C\notin X$. If there was an FD $X\to C\in\Sigma'$ and $C\in X$, then

$$(\Sigma'-\{X\to C\})\cup\{X-C\to C\}$$

is equivalent to Σ but contains less attributes than Σ'.

Next we prove that there is no cover Σ' of Σ with respect to R_s with a smaller number of attribute occurrences. Suppose there were an optimal cover Σ' of Σ with respect to R_s with a fewer number of attribute occurrences than Σ. Then for all σ' in Σ' it is the case that $\Sigma\models_{R_s}\sigma'$. Consequently, there must be some $\sigma\in\Sigma$ such that $LHS(\sigma)\subseteq LHS(\sigma')$. Suppose every FD $\sigma\in\Sigma$ has the property that $LHS(\sigma)\subseteq LHS(\sigma')$ for a different FD $\sigma'\in\Sigma'$. Then Σ' contains at least as many attribute occurrences as Σ, a contradiction. Otherwise, there is a proper subset Σ'' of Σ such that every FD $\sigma'\in\Sigma'$ has the property that $LHS(\sigma)\subseteq LHS(\sigma')$ for some $\sigma\in\Sigma''$. Consequently, Σ'' implies every FD in Σ' with respect to R_s and therefore also every FD in Σ. This, however, is impossible since Σ is non-redundant.

Thus we have just shown that Σ is its own optimal cover with respect to R_s, and thus exponential in the number of attributes. Now we show that there is an Armstrong table for Σ and R_s where the number of tuples is in $\mathcal{O}(n)$. It suffices to show that the set $max_{\Sigma,R_s}(R)$ contains a number of elements that is linear in the number of attributes. For each $i=1,\ldots,n$ we have $max_{\Sigma,R_s}(A_i)=R-A_i$, and $max_{\Sigma,R_s}(B_i)=R-B_i$. These are $2n$ different maximal sets in total. The set $max_{\Sigma,R_s}(C)$ consists of the following n elements: $R-A_iB_iC, i=1,\ldots,n$. Therefore, $max_{\Sigma,R_s}(R)$ has $3n$ different elements. Using Mannila and Räihä's algorithm [96] (which applies since we are in the case where $R_s=R$) we can easily create an Armstrong table for Σ and R_s that has $3n+1$ tuples only. □