Design by Example for SQL Tables with Functional Dependencies

Sven Hartmann · Markus Kirchberg · Sebastian Link

Received: date / Accepted: date

Abstract An Armstrong relation satisfies the functional dependencies (FD) implied by a given FD set and violates all other FDs. Therefore, they form an instance of the design-by-example methodology: the example relation exhibits precisely those FDs that the current design perceives as meaningful for the application domain. The existing framework of Armstrong relations only applies to total relations.

We investigate structural properties of Armstrong relations for FDs over SQL tables. That is, arbitrary attributes can be declared NOT NULL and null values apply Zaniolo’s most general no information interpretation. In contrast to total relations, FDs do not enjoy Armstrong relations. However, the class of standard FDs with arbitrary NOT NULL constraints does enjoy Armstrong relations. While the problem of finding an Armstrong relation is shown to be precisely exponential, we establish an algorithm that computes Armstrong relations with a size at most quadratic in that of a minimum-sized Armstrong relation. Our results enable designers to make effective use of Armstrong relations for real SQL tables. This can lead to real world designs that guarantee efficient ways of data processing.

Keywords SQL · Schema design · Functional dependency · Armstrong relation

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 [22]. In this model, data is 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. The relation schema itself is a set of attributes which represent the properties that each tuple of each relation over the schema is described by. That is, a tuple maps each attribute of the relation schema to a value from the domain of this attribute. Relations permit the storage of inconsistent data, i.e., data that violate conditions which every meaningful relation ought to satisfy. Consequently, additional assertions, called data dependencies, are specified by the data administrator in order to restrict the relations to those which are considered meaningful to the application at hand. Functional dependencies (FDs) form a very important class of data dependencies. According to [29] they make up around two-thirds of all uni-relational data dependencies (dependencies defined over a single relation schema) in practice. They are essential in database modeling, design, normalization and maintenance [1], and have also proven application in data integration [19,78], semantic query optimization [20,35] and database security [15] to name a few. Among many challenges, the database design team must recognize the set of FDs that are meaningful for the underlying application domain.

Example 1 Suppose that in designing an information system for a company the design team 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. An SQL definition of the table may look as follows:

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

Moreover, the design team has recognized additional constraints that the database management system will enforce on any relations over 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 two columns. This is expressed in the form of NOT NULL constraints on the attributes of the SQL table above. Secondly, every employee can only work in one department. Therefore, the design team recommends the specification of the functional dependency Emp → Dept. Finally, every department has only one manager. Consequently, the team also recommends to enforce the functional dependency Dept → Mgr.

One obstacle in the challenge to recognize the set of meaningful FDs is the difficulty to fully comprehend their interaction. That is, the explicit enforcement of a set of FDs usually means that some other FDs are implicitly enforced as well. It is therefore an important task to assist design teams in their understanding of these interactions. More formally, given a set $\Sigma$ of FDs, we say that $\Sigma$ implies $\phi$ if every relation that satisfies all FDs in $\Sigma$ also satisfies $\phi$. We use $\Sigma^*$ to denote the semantic closure of $\Sigma$, i.e., the set of all FDs implied by $\Sigma$. Armstrong relations have been identified as an effective means to represent concisely abstract sets of FDs in the form of a single relation. More precisely, an Armstrong relation for a set $\Sigma$ of FDs is a single relation that satisfies every FD in $\Sigma^*$ and violates every FD not in $\Sigma^*$ [40]. Hence, an FD is implicit in the explicit specification of an FD set $\Sigma$ if and only if it is satisfied by an Armstrong relation for $\Sigma$. For this reason, Armstrong relations represent one of the few instances where example-based reasoning is effective. That is, if $\Sigma$ constitutes the design choice of the set of FDs currently perceived as meaningful to the underlying application domain, then an Armstrong relation $r_\Sigma$ for $\Sigma$ constitutes an example on which the design team can test the meaningfulness of an arbitrary functional dependency $\phi$. Namely, $\phi$ is meaningful under the current design choice if and only if the example relation $r_\Sigma$ satisfies $\phi$.

Consequently, this approach to design is called design-by-example. Indeed, Armstrong relations are widely regarded as a helpful tool for design participants to judge, justify, convey, or test their understanding of the relation schema [27,75]. Recently, empirical evidence has been presented that Armstrong relations can help design teams to identify actually meaningful FDs that they have perceived as meaningless before inspecting an Armstrong relation [64]. We exemplify these observations by the following case.

**Example 2** Consider the relation schema EMPLOYMENT with attributes Emp, Dept, and Mgr, and FD set $\Sigma$ that consists of $\text{Emp} \rightarrow \text{Dept}$ and $\text{Dept} \rightarrow \text{Mgr}$. Suppose for now that every relation over EMPLOYMENT is total, or equivalently, that all three attributes are declared NOT NULL. In this case, the relation

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilbert</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Pythagoras</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Einstein</td>
<td>Physics</td>
<td>Gauss</td>
</tr>
<tr>
<td>Turing</td>
<td>Computer Science</td>
<td>von Neumann</td>
</tr>
</tbody>
</table>

is an Armstrong relation for $\Sigma$ [9]. An inspection of this Armstrong relation tells the design team that they have captured the meaningful constraint that the manager is uniquely determined by the employee, i.e. that the FD $\text{Emp} \rightarrow \text{Mgr}$ is implied by $\Sigma$. In other words, Emp forms a minimal key for EMPLOYMENT. An inspection of the Armstrong relation also shows that the FD $\text{Mgr} \rightarrow \text{Dept}$ is not implied by $\Sigma$: the design team simply notices that Gauss is the manager of both the Math and Physics department. Consequently, this FD is not yet enforced implicitly by enforcing explicitly all FDs in $\Sigma$. Therefore it must be specified explicitly, if it represents a meaningful business rule.

Surprisingly, most of the theory of Armstrong relations only applies to FDs over total relations, i.e., where null values are not permitted to occur at all. Therefore, design teams have no effective support in acquiring the set of meaningful FDs over SQL tables where arbitrary attributes can be specified as NOT NULL. While FDs have been studied in the context of such constraints [6,69], the concept of Armstrong relations has not been investigated yet. To address this research gap, we adopt the formal framework of Lien [69], 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 this observation, and motivates the study of Armstrong relations for real SQL table definitions.

**Example 3** Consider the relation schema **EMPLOYMENT** with attribute set **Emp**, **Dept** and **Mgr** and FD set Σ that consists of **Emp → Dept** and **Dept → Mgr**. As in Example 1 assume that only **Emp** and **Mgr** have been declared NOT NULL. In this case, the relation

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilbert</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Pythagoras</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Einstein</td>
<td>Physics</td>
<td>Gauss</td>
</tr>
<tr>
<td>Turing</td>
<td>ni</td>
<td>von Neumann</td>
</tr>
<tr>
<td>Turing</td>
<td>ni</td>
<td>Gödel</td>
</tr>
</tbody>
</table>

is an Armstrong relation for Σ and the NOT NULL constraints. Note that, according to the semantics under the no information interpretation [6, 69, 57], the relation satisfies the FD **Dept → 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. An inspection of this Armstrong relation reveals to the design team that the FD **Emp → Mgr** is not implied by Σ and the NOT NULL constraints: Turing has managers von Neumann and Gödel. Specifically, it follows also that **Emp** is not a key for **EMPLOYMENT**. If these FD and key represent meaningful semantic constraints, then the design team has now a choice of either specifying the FD **Emp → Mgr** additionally to the FDs in Σ, or declaring **Dept** as NOT NULL.

Contributions and Organization. In this article we investigate the concept of Armstrong tables for FDs in the presence of NOT NULL constraints. Following previous work [6, 57, 69], we adopt Zaniolo, Lien, Atzeni and Morfuni’s no information interpretation of null values. This is the most primitive interpretation which allows design teams 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 in SQL table definitions.

The objective of this article is to extend the existing toolbox of Armstrong relations such that it becomes applicable to real SQL table definitions with FDs. We review previous work in Section 2, and define the underlying concepts and present related previous results in Section 3.

As a first contribution we show in Section 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 $\emptyset \to A$ 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 [43]. 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 Section 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 [75] 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 Section 6 an algorithm that, given an arbitrary set $\Sigma$ of standard FDs and an arbitrary NFS $R_s$ over an arbitrary relation schema $R$, computes an Armstrong table for $\Sigma$ and $R_s$. The algorithm is a non-trivial extension of the algorithm by Mannila and Räihä for FDs over total relations [75]. The main part of the algorithm 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. However, 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 Section 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 $\Sigma$. Therefore, just for the reason of the size of a representation, design teams 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 Section 8 how our results carry over to Armstrong tables for standard FDs and an NFS in the world of all FDs. In contrast to the world of standard FDs, Armstrong tables for the world of all FDs must additionally violate all non-trivial non-standard FDs.

As a final contribution we illustrate in Section 9 the impact of our results on database practice. We show how Armstrong tables can assist design teams in recognizing meaningful constraints that were previously incorrectly perceived as meaningless. We demonstrate how 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 [5,6,9,30,69,75] are retained in the more general case. We conclude in Section 10 where we also discuss multiple directions of future work.

2 Related Work

Data dependencies have been studied thoroughly in various data models, and we refer the reader to more detailed surveys [1,44,73,84].

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 [22,23]. A total relation over a relation schema $R$ satisfies a key $X$ if and only if it satisfies the functional dependency $X \rightarrow R$. 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 database designers 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 $\Sigma$ implies $\varphi$. For FDs over total relations, implication can be decided in time linear in the total number of attributes that occur in the input [7,37]. 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 towards 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 [7,8,11,14,23], requirements engineering and schema validation [75], data mining [76], in database security [13,15], view maintenance [62] and in query optimization [20,35,36]. More recently, the implication
problem has received a lot of attention in other data models as well [3, 4, 6, 17, 18, 47, 50, 54–56, 60, 63, 65, 82, 83, 88–91]. New application areas involve data cleaning [45], data transformations [26], consistent query answering [21] and data exchange [42, 67, 77] and data integration [19, 78].

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 [30, 31, 61] and FDs [5, 9, 34, 75]. In particular, the existence of Armstrong relations for FDs was shown by Armstrong [5], and Fagin [41] has shown the existence of Armstrong relations for a wide class of data dependencies; however, there exist classes of data dependencies that do not enjoy Armstrong relations. The structure of Armstrong relations for sets of FDs over total relations has mainly been investigated by Beeri, Fagin, Statman and Howard [9] and Mannila and R"aih"a [75]. In the 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 [28, 33, 40, 43, 49, 75]. An excellent survey on Armstrong databases is [40]. Recently, the concept of informative Armstrong databases was introduced [28]. These are small subsets of an existing database, satisfying exactly the same data dependencies. Note that design aids [27, 75, 81] use Armstrong databases to help judge, justify, convey, or test the database designer team’s understanding of a given relation schema. Recently, empirical evidence has been established that Armstrong relations help design teams to recognize meaningful FDs that they were not able to recognize without the help of Armstrong relations [64].

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 [22] is incomplete information [25, 58, 66]. 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, or-relations [59, 68, 86] or fuzzy relations [82]. 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” [24, 52, 53], “non-existence” [74], “inapplicable” [53], “no information” [92] and “open” [51]. 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 [69] 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 [57] 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 [80]. Levene and Loizou introduced and axiomatized the classes of weak and strong FDs with respect to a possible world semantics [65]. The axiomatization of strong FDs is given by the Armstrong axioms, while weak FDs have the same axiomatization as the FDs of Lien [69], 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 → Dept since the null value ni can be replaced by Comp (there is a possible world in which Emp → Dept is satisfied). However, under the no information interpretation the functional dependency Emp → 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 [65].

Over the last decade, FDs have also received a lot of attention from the XML community [4, 56, 63, 88, 91]. However, the study of Armstrong data trees for XML FDs has only been preliminary so far. In fact, Hartmann et al. [56] 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 functional dependencies over SQL
tables. Our development will follow the extension of Codd’s relational model of data [22] to encompass incomplete information by the no information null value introduced by Zaniolo [92], applied to functional dependencies by Lien [69], to functional dependencies and null existence constraints by Atzeni and Morfuni [6], and to functional and multivalued dependencies and a null-free subschema by Hartmann and Link [57].

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 \( \text{dom}(A) \) 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 \( \text{ni} \in \text{dom}(A) \).

In the literature, many kinds of null values have been proposed; for example, “missing” or “value unknown at present” [24, 53, 52], “non-existence” [74], “inapplicable” [53], “no information” [92] and “open” [51]. The intention of the null value \( \text{ni} \) is to mean “no information”. That is, the null value \( \text{ni} \) 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 \( t : R \rightarrow \bigcup_{A \in R} \text{dom}(A) \) with \( t(A) \in \text{dom}(A) \) 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 \( \text{dom}(X) = \prod_{A \in X} \text{dom}(A) \) 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 \( t_2[A] = \text{ni} \). In line with previous work [6, 69, 92, 70], 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.

For a set \( X \subseteq R \), a tuple \( t \) over \( R \) is \( X \)-total, if for all \( A \in X \), \( t[A] \neq \text{ni} \). 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 FDs and Null-Free Subschemata

Functional dependencies between sets of attributes have always played a central role in the study of relational databases [7, 11, 22], and seem to be central for the study of database design in other data models as well [4, 54, 63, 65, 83, 85, 88–90]. 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].

According to Lien [69], a functional dependency (FD) over the relation schema \( R \) is a statement \( X \rightarrow Y \) where \( X, Y \subseteq R \). The FD \( X \rightarrow Y \) over \( R \) is satisfied by a relation \( r \) over \( R \), denoted by \( r \models X \rightarrow Y \), if and only if for all \( t_1, t_2 \in r \) the following holds: if \( t_1 \) and \( t_2 \) are \( X \)-total and \( t_1[X] = t_2[X] \), then \( t_1[Y] = t_2[Y] \). 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. FDs of the form \( \emptyset \rightarrow Y \) are called non-standard, otherwise they are called standard.

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. In fact, tuples with nulls in attributes in \( X \) cannot cause a violation of a functional dependency \( X \rightarrow Y \): the nulls mean that no information is available about those attributes. On the other hand, two \( X \)-total tuples \( t_1, t_2 \) where \( t_1[X] = t_2[X] \) and \( t_2[A] \) is \( A \)-total while \( t_1[A] \) is not, violate any dependency \( X \rightarrow Y \) with \( A \in Y \): the tuple \( t_1 \) indicates that no information is available about the value for \( A \) associated with \( t_1[X] \), while the tuple \( t_2 \) indicates that the value for \( A \) associated with \( t_2[X] = t_1[X] \) does exist. Hence, it violates the natural requirement of a functional dependency that if the values for \( X \) are the same for two tuples, both tuples must contain the same information for the attributes in \( Y \).

According to Atzeni and Morfuni [6], a null-free subschema (NFS) over the relation schema \( R \) is an expression \( R_s \) such that \( R_s \subseteq R \). The NFS \( R_s \) over \( R \) is satisfied by a partial relation \( r \) over \( R \), denoted by \( r \models R_s \), if and only if for all \( t \in r \) and for all \( A \in R_s \) we have \( t[A] \neq \text{ni} \). 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 $\Sigma$ of constraints over some relation schema $R$, we say that a (partial) relation $r$ over $R$ satisfies $\Sigma$, denoted by $|_r \Sigma$, if $r$ satisfies every element of $\Sigma$. If for some $\sigma \in \Sigma$ the relation $r$ does not satisfy $\sigma$ we sometimes say that $r$ violates $\sigma$ (in which case $r$ also violates $\Sigma$) and write $\not |_r \sigma$ ($\not |_r \Sigma$).

Example 4 For Example 1 we obtain the relation schema

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

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

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing</td>
<td>n1</td>
<td>von Neumann</td>
</tr>
<tr>
<td>Turing</td>
<td>n1</td>
<td>Gödel</td>
</tr>
</tbody>
</table>

satisfies $\text{Employment}_s$ as well as the FDs $\text{Emp} \to \text{Dept}$ and $\text{Dept} \to \text{Mgr}$. However, the relation violates the FD $\text{Emp} \to \text{Mgr}$.

3.3 Implication and Inference

For the design of a relational database schema dependencies are normally specified as semantic constraints on the relations which are intended to be instances of the schema. During the design process or the 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 $C$: functional dependencies 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 $\Sigma$ implies $\varphi$ in the presence of $R_s$, denoted by $\Sigma \vdash_{R_s} \varphi$, if every relation $r$ over $R$ that satisfies $\Sigma$ and $R_s$ also satisfies $\varphi$. If $\Sigma$ does not imply $\varphi$ in the presence of $R_s$, we may also write $\Sigma \not \vdash_{R_s} \varphi$.

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 $\Sigma \vdash_{R_s} \varphi$. For the class of FDs in the presence of an NFS, the sets $\Sigma \cup \{\varphi\}$ for a relation schema $R$ are always finite. Moreover, if $R_s = \emptyset$ we also write $\Sigma \vdash \varphi$ instead of $\Sigma \vdash_{\emptyset} \varphi$. This covers the case where every attribute is NULL [69]. 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 $\Sigma$ over a relation schema $R$ and an NFS $R_s$ over $R$, let the FD set $\Sigma^*_R = \{\varphi \mid \Sigma \vdash_{R_s} \varphi\}$ denote the semantic closure of $\Sigma$. 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 $\varphi$ from $\Sigma$ by $\mathfrak{R}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of FDs such that $\sigma_n = \varphi$ and every $\sigma_i$ is an element of $\Sigma$ 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 $\Sigma$ let $\Sigma^+_R = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ denote the syntactic closure of $\Sigma$ 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 $\Sigma$ over $R$, we have $\Sigma^+_R \subseteq \Sigma^*_R$. 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 5 Consider the relation schema $\text{Employment}$ with the NFS $\text{Employment}_s$ from Example 4 again. Let $\Sigma$ denote the set of FDs over $\text{Employment}$ that consists of $\text{Emp} \to \text{Dept}$ and $\text{Dept} \to \text{Mgr}$. Since the relation in Example 4 satisfies the NFS $\text{Employment}_s$ as well as $\Sigma$, but it violates the FD $\text{Emp} \to \text{Mgr}$ we conclude that $\text{Emp} \to \text{Mgr}$ is not in the semantic closure $\Sigma^*_\text{Employment}_s$ of $\Sigma$.

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.

![Table 1](image)

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

$$X \to Y$$

follows from the reflexivity axiom and the null transitivity rule [6].
Consider the relation schema Employment with the NFS Employment, from Example 4 again. Then the FD $\text{Emp} \rightarrow \text{Mgr}$ cannot be inferred from the two FDs $\text{Emp} \rightarrow \text{Dept}$ and $\text{Dept} \rightarrow \text{Mgr}$ by means of the null transitivity rule since the attribute Dept is not an element of Employment.

Atzeni and Morfuni also established a linear-time algorithm for deciding the implication problem for functional dependencies 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 $X^\Sigma_{
exists R_r} = \{ A \in R | \Sigma \models_{R_r} X \rightarrow A \}$ of an attribute set $X$ with respect to an FD set $\Sigma$ and an NFS $R_r$ over the relation schema $R$ [6]. An FD $X \rightarrow Y$ over $R$ is implied by $\Sigma$ in the presence of $R_r$ if and only if $Y \subseteq X^\Sigma_{
exists R_r}$ holds [6]. Algorithm 1 computes the attribute closure $X^\Sigma_{
exists R_r}$ of $X$ with respect to $\Sigma$ and $R_r$ over $R$ [6].

Algorithm 1 (NFSClosure($X, \Sigma, R_r, r$))

Input: attribute subset $X$, FD set $\Sigma$, NFS $R_r$ all over $R$, relation schema $R$

Output: attribute closure $X^\Sigma_{
exists R_r}$ of $X$ with respect to $\Sigma$ and $R_r$

Method:

(A0) CLOSURE := $X$;

(A1) OLD CLOSURE := $\emptyset$;

(A2) while CLOSURE $\neq$ OLD CLOSURE do

OLD CLOSURE := CLOSURE;

for all $V \rightarrow W \in \Sigma$ do

if $V \subseteq \text{CLOSURE} \cap X R_r$, then

CLOSURE := CLOSURE $\cup$ W;

endif;

enddo;

enddo;

(A3) return CLOSURE;

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

$$\{\text{Emp}, \Sigma, \{\text{Dept}\}, \text{Employment}\}$$

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

Example 7 Consider the relation schema Employment with the NFS Employment, from Example 4 again, and let $\Sigma$ consist of the two FDs $\text{Emp} \rightarrow \text{Dept}$ and $\text{Dept} \rightarrow \text{Mgr}$. On input

$$\{\text{Emp}, \Sigma, \text{Employment}, s, \text{Employment}\}$$

Algorithm 1 returns the set $\{\text{Emp, Dept}\}$. Consequently, the FD $\text{Emp} \rightarrow \text{Mgr}$ is not implied by $\Sigma$ in the presence of the NFS Employment since Mgr is not an element of $\{\text{Emp}\}_{s, \text{Employment}}^\Sigma$.

3.4 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 $\text{unique}(X)$ with $X \subseteq R$. A relation $r$ over $R$ satisfies $\text{unique}(X)$ if and only if for all different $t, t' \in r$ the following holds: if $t$ and $t'$ are X-total, then $t[X] \neq t'[X]$. It follows that $r$ satisfies $\text{unique}(X)$ if and only if $r$ satisfies the FD $X \rightarrow R$.

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

It is now relatively easy to see that we can express uniqueness constraints and Codd keys by means of functional dependencies and a null-free subschema.

Theorem 2 Let $\Sigma$ be a set of FDs and let $R_r$ be an NFS over $R$. Then the following hold:

1. $\Sigma \models_{R_r} \text{unique}(X)$ if and only if $\Sigma \models_{R_r} X \rightarrow R$,

2. $\Sigma \models_{R_r} \text{Codd}(X)$ if and only if $\Sigma \models_{R_r} X \rightarrow R$ and $X \subseteq R_r$.

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

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

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

Suppose now that $X \not\subseteq R_r$ holds, i.e., $X \not\subseteq R_r$ 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 $A \in R - R_r$ and $t[B] \in \text{dom}(B) - \{ni\}$ for all $B \in R_r$. It follows that $r$ satisfies $\Sigma$ and the NFS $R_r$, but $r$ violates $\text{Codd}(X)$ since $X \cap (R - R_r) \neq \emptyset$. 


3.5 Armstrong relations

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

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

In this paper, we are interested in the class $C$ 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 8 The relation of Example 2 is an Armstrong table for the FD set

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

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

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 Section 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 a characterization will be established in Section 5. The characterization will assist us in Section 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 Section 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 functional dependencies enjoy Armstrong relations $[5,9,40]$. We will show now, however, that the class of functional dependencies does not enjoy Armstrong tables if there are attributes that are not declared NOT NULL. Intu-}

itively in such a case, non-standard functional dependencies 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 $\Sigma$ over $R$ such that there is no Armstrong table for $\Sigma$ and $R_s$.

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

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c'$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ni</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>ni</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

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

We show now that any relation that satisfies the FD $\emptyset \rightarrow A$ and violates the FDs $AB \rightarrow C$ and $C \rightarrow B$ must violate the FD $A \rightarrow B$. Let $r$ be a relation that satisfies the FD $\emptyset \rightarrow A$ and violates the FDs $AB \rightarrow C$ and $C \rightarrow B$. Since $r$ violates $AB \rightarrow C$ there are two distinct tuples $t_1, t_2 \in r$ such that $\text{t}_1[AB] = \text{t}_2[AB]$ and $\text{t}_1, \text{t}_2$ are both $AB$-total, and $\text{t}_1[C] \neq \text{t}_2[C]$. Since $r$ violates the FD $C \rightarrow B$ there are two distinct tuples $t_3, t_4 \in r$ such that $\text{t}_3[C] = \text{t}_4[C]$ and $\text{t}_3, \text{t}_4$ are both $C$-total, and $\text{t}_3[B] \neq \text{t}_4[B]$. Since $r$ satisfies the FD $\emptyset \rightarrow A$, it follows that $\text{t}_3[A] = \text{t}_4[A] = \text{t}_1[A]$, and, in particular, that $\text{t}_3$...
and $t_4$ are $A$-total. Hence, the tuples $t_3, t_4$ witness that $r$ violates the FD $A \rightarrow B$. Consequently, $r$ cannot be an Armstrong table for $\Sigma$ and $R_s$. □

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

However, the proof of Theorem 3 also shows that Lien’s class of FDs over partial relations [69] does not enjoy Armstrong relations. That is, the proof argument also applies when no NFS $R_s$ is present. In particular, note that $\Sigma \not\models AB \rightarrow C$ and $\Sigma \not\models C \rightarrow B$ 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 9) and the general case (Example 10) we consider here.

**Example 9** Consider the relation schema

\[
\text{EMPLOYMENT}=\{\text{Emp, Dept, Mgr}\},
\]

the NFS $R_s = \{\text{Emp, Dept, Mgr}\}$ and let $\Sigma$ consist of the two FDs $\emptyset \rightarrow \text{Mgr}$ and $\text{Mgr} \rightarrow \text{Dept}$. The relation $r$ is an Armstrong table for $\Sigma$ and $R_s$. In particular, note that the FD $\emptyset \rightarrow \text{Dept}$ is implied by $\Sigma$ and $R_s$. □

**Example 10** Consider the relation schema

\[
\text{EMPLOYMENT}=\{\text{Emp, Dept, Mgr}\},
\]

the NFS $R_s = \{\text{Emp, Dept}\}$ and let $\Sigma$ consist again of the two FDs $\emptyset \rightarrow \text{Mgr}$ and $\text{Mgr} \rightarrow \text{Dept}$. This is the case of the proof in Theorem 3. Here, the relation $r'$ is not an Armstrong table for $\Sigma$ and $R_s$. Indeed, it violates all FDs not implied by $\Sigma$ and $R_s$, and satisfies all FDs implied by $\Sigma$ and $R_s$ except $\emptyset \rightarrow \text{Mgr}$. □

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 [43].

## 5 Characterization of Armstrong Tables

In this section we continue Lien [69], Atzeni and Morfoni’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, 75]. Our generalization requires an extension of Demetrovic’s notion of maximal sets of attributes [30, 75] and a refinement of Mannila, Räihä, Beeri, Dowd, Fagin and Statman’s notion of an agree set of tuples. Due to Theorem 3 we will focus in Sections 5, 6 and 7 on the class $\mathcal{C}$ of standard FDs in the presence of an NFS, before we return to the class of all FDs in the presence of an NFS in Section 8.

### 5.1 Agree Sets and Maximal Families of Sets

**Definition 1** Let $\Sigma$ 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 $\Sigma$ with respect to $\Sigma$ and $R_s$ as follows:

\[\max_{\Sigma,R_s}(A) := \{\emptyset \subseteq X \mid \Sigma \not\models R_s, \Sigma \models X \land \forall B \in C (X \models R_s, XB \rightarrow A)\}.\]

The **maximal sets** of $R$ with respect to $\Sigma$ and $R_s$ are defined as $\max_{\Sigma,R_s}(R) = \bigcup_{A \in R} \max_{\Sigma,R_s}(A)$. If $\Sigma$ 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 $\Sigma$ and $R_s$ are the maximal attribute subsets of the underlying relation schema $R$ that do not functionally determine $A$.

**Example 11** Let $\Sigma = \{\text{Emp} \rightarrow \text{Dept}, \text{Dept} \rightarrow \text{Mgr}\}$ and let $R_s = \{\text{Emp, Dept, Mgr}\}$ be an NFS over the relation schema EMPLOYMENT. Then $\max_{\Sigma,R_s}(\text{Emp})$ consists of $\{\text{Dept, Mgr}\}$, $\max_{\Sigma,R_s}(\text{Dept})$ contains the single element $\{\text{Mgr}\}$, and $\max_{\Sigma,R_s}(\text{Mgr}) = \emptyset$.

For $R_s = \{\text{Emp, Mgr}\}$, $\max_{\Sigma,R_s}(\text{Emp})$ contains the single element $\{\text{Dept, Mgr}\}$, $\max_{\Sigma,R_s}(\text{Dept})$ contains the single element $\{\text{Mgr}\}$, and $\max_{\Sigma,R_s}(\text{Mgr})$ contains the singleton $\{\text{Emp}\}$. □
Definition 2  Let $R$ be some relation schema. For two tuples $t, t'$ over $R$ let

$$
ag(t, t') := \{(X,Y) \mid \forall A \in R (\{(t[A] = t'[A] \land [A] \neq \text{ni}) \rightarrow A \in X\) \land (t[A] = t'[A] \rightarrow A \in Y)) \}.
$$

denote the agree set of $t$ and $t'$. For a relation $r$ over $R$ let

$$
ag(r) = \{ag(t, t') \mid t, t' \in r \land t \neq t'\}
$$

denote the agree set of $r$, let

$$
ag^w(r) = \{X \mid (X, Y) \in ag(r)\}
$$

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 $X \in ag^w(r)$ let

$$
w(X) = \bigcap\{Y \mid (X, Y) \in ag(r)\}.
$$

Remark 1  Over total relations [9,75], it suffices to define $ag(t, t') := \{A \in R \mid t[A] = t'[A]\}$ since a total relation $r$ satisfies $ag^w(r) = ag^w(r)$, and $w(X) = X$ for all $X \in ag(r)$. Note that all elements of $ag(r)$ have the form $(X, X)$ in the case of total relations.

Example 12  Let $r$ denote the relation from Example 3:

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilbert</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Pythagoras</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Einstein</td>
<td>Physics</td>
<td>Gauss</td>
</tr>
<tr>
<td>Turing</td>
<td>ni</td>
<td>von Neumann</td>
</tr>
<tr>
<td>Turing</td>
<td>ni</td>
<td>Gödel</td>
</tr>
</tbody>
</table>

The agree set of the last two tuples is

$$
\{\text{Emp}, \text{Emp, Dept}\}
$$

and the remaining agree sets of $r$ are:

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

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

5.2 A First Characterization

For a relation $r$ over a relation schema $R$ let

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

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

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

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

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

and

2. total$(r) = R_*$.

Proof Sufficiency. Let $r$ be a relation over $R$ that satisfies the conditions. We show that $r$ is an Armstrong table for $\Sigma$ and $R_*$. Let $X \rightarrow A \in \Sigma$. That is, $A \in X^*_{\Sigma, R_*}$. Assume that there are distinct tuples $t, t' \in r$ such that $t[A] = t'[A]$ and $t, t'$ are $X$-total. That is, $X \subseteq X' = ag^w(t, t')$. 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 $\Sigma$.

Let $X \rightarrow A \not\in \Sigma_{R_*}$. Hence, $A \not\in X^*_{\Sigma, R_*}$. By the first condition there is some $Z \in ag^w(r)$ such that $X \subseteq Z \land A \not\in w(Z)$. In particular, there are tuples $t, t'$ such that $X \subseteq Z = ag^w(t, t')$ and $A \not\in ag^w(t, t')$. 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 $\Sigma_{R_*}$.

Finally, the condition $\text{total}(r) = R_*$ ensures that $r$ satisfies every NFS implied by $R_*$ and violates every NFS not implied by $R_*$, Consequently, $r$ is indeed an Armstrong table for $\Sigma$ and $R_*$.

Necessity. Let $r$ be a relation over $R$ that is Armstrong for $\Sigma$ and $R_*$. We show that $r$ satisfies the conditions.

Let $t, t' \in r$ be distinct tuples such that $X \subseteq X' = ag^w(t, t')$. As $r$ satisfies $\Sigma_{R_*}$, we have $X^*_{\Sigma, R_*} \subseteq ag^w(t, t')$, and thus $X^*_{\Sigma, R_*} \subseteq w(X')$. Therefore, $X^*_{\Sigma, R_*} \subseteq \bigcap\{ w(Z) \mid X \subseteq Z \in ag^w(r)\}$ holds.

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

Finally, since $r$ is Armstrong for $\Sigma$ and $R_*$ it follows that $\text{total}(r) = R_*$.

5.3 A Second Characterization

For the special case of total relations, i.e. where $R_* = R$, it is well-known that every FD set $\Sigma$ and the NFS $R$ over $R$ defines a closure operator $(\cdot)^*_{\Sigma, R} : \mathcal{P}(R) \rightarrow \mathcal{P}(R)$
by mapping every attribute subset $X \subseteq R$ to its attribute closure $X^*_{\Sigma, R}$. In fact, $(\cdot)^*_{\Sigma, R}$ 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 ($(X^*_{\Sigma, R})^*_{\Sigma, R} = X^*_{\Sigma, R}$). In particular, an attribute subset $X$ of $R$ is said to be closed with respect to the FD set $\Sigma$ 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 $\Sigma$ and the NFS $R$ is denoted by $cl_{\Sigma, R}(R)$. For the special case where $R_s = R$ it is well-known that $cl_{\Sigma, R}(R)$ 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, Dowd, Fagin, and Statman [9] have shown that $r$ is an Armstrong table for an FD set $\Sigma$ and the NFS $R$ over $R$ if and only if $gen_{\Sigma, R}(R) \subseteq ag^*(r) \subseteq cl_{\Sigma, R}(R)$ holds. Later, Mannila and Räihä [75] have shown that for an arbitrary relation schema $R$, an arbitrary FD set $\Sigma$ 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, $(\cdot)^*_{\Sigma, R_s}$ : $\mathcal{P}(R) \rightarrow \mathcal{P}(R)$ 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 3 where the FD set $\Sigma$ consists of the two FDs $Emp \rightarrow Dept$ and $Dept \rightarrow Mgr$, and the NFS $R_s$ consists of the attributes $Emp$ and $Mgr$. Then the attribute subset closures are:

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

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 $cl_{\Sigma, R_s}$ 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 $\Sigma$ and a given NFS $R_s$ over the relation schema $R$.

**Theorem 5** Let $R$ be some relation schema, let $\Sigma$ 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 $\Sigma$ and $R_s$ if and only if the following conditions are satisfied:

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

**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 $\Sigma$ and $R_s$.

Let $X \rightarrow A \in \Sigma$. Assume that there are distinct $t, t' \in r$ such that $t[X] = t'[X]$ and $t$ is $X$-total. That is, $X \subseteq X' = ag^*(t, t')$. 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, $A \in ag^*(t, t')$. Therefore, $t[A] = t'[A]$. Hence, $r$ satisfies $\Sigma$.

Let $X \rightarrow A \notin \Sigma_{R_s}$. It follows that there is $X' \in max_{\Sigma, R_s}(A)$ such that $X \subseteq X'$ and $A \notin (X')^*_{\Sigma, R_s}$. Condition 1. implies that $X' \in ag^*(r)$ and $A \notin w(X')$. Hence, there is some $Y \in ag^*(r)$ 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 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 $\Sigma$ and $R_s$.

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

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

Next we show that $r$ satisfies condition 2. Therefore, let $X \in ag^*(r)$. 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 $Y \in ag^*(r)$ such that $(X, Y) \in ag(r)$ and $A \notin Y$. Consequently, there are some distinct $t, t' \in r$ such that $X = ag^*(t, t')$ and $A \notin ag^*(t, t')$. That is, $r$ violates the FD $X \rightarrow A$. Since $r$ is an Armstrong table for $\Sigma$ 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 \( \Sigma \) and \( R_s \) it follows that total(\( r \)) = \( R_s \).

Remark 2  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 \( X \in \text{ag}(r) \) (and that \( A \notin X \) which follows from \( X \in \max_{\Sigma, R_s}(A) \)). That is, condition 1. says that \( \max_{\Sigma, R_s}(R) \subseteq \text{ag}(r) \). Furthermore, condition 2. says that \( \forall X \in \text{ag}(r) \forall X_s \subseteq X \), i.e., all (strong) agree sets are closed: \( \text{ag}(r) \subseteq \text{cl}_{\Sigma, R_s}(R) \). □

The following example illustrates Theorem 5.

Example 14 Let Employment and the FD set
\[ \Sigma = \{\text{Emp} \rightarrow \text{Dept}, \text{Dept} \rightarrow \text{Mgr}\} \]
as before, but let \( R_s = \{\text{Emp, Mgr}\} \). The following relation \( r \):

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilbert</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Pythagoras</td>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>ni</td>
<td>Astronomy</td>
<td>Newton</td>
</tr>
<tr>
<td>ni</td>
<td>Physics</td>
<td>Newton</td>
</tr>
<tr>
<td>Turing</td>
<td>ni</td>
<td>von Neumann</td>
</tr>
<tr>
<td>Turing</td>
<td>ni</td>
<td>G ö del</td>
</tr>
</tbody>
</table>

is an Armstrong table for \( \Sigma \) and \( R_s \). Indeed, the maximal set families are:
- \( \max_{\Sigma, R_s}(\text{Emp}) = \{\{\text{Dept, Mgr}\}\} \),
- \( \max_{\Sigma, R_s}(\text{Dept}) = \{\{\text{Mgr}\}\} \), and
- \( \max_{\Sigma, R_s}(\text{Mgr}) = \{\{\text{Emp}\}\} \).

The agree sets of \( r \) are:
- \( (\emptyset,\emptyset) \),
- \( (\{\text{Dept, Mgr}\}, \{\text{Dept, Mgr}\}) \),
- \( (\{\text{Mgr}\}, \{\text{Mgr}\}) \),
- \( (\{\text{Emp}\}, \{\text{Emp, Dept}\}) \).

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

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 \( \Sigma \), and relation \( r \) be given as in Example 2, but let the NFS \( R_s \) be \{\text{Emp, Mgr}\}. Recall from Example 11 the maximal sets \{\text{Dept, Mgr}\}, \{\text{Mgr}\} and \{\text{Emp}\}. The agree sets of \( r \) are:

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

While condition 2. of Theorem 5 is satisfied, conditions 1. and 3. are both violated. In fact, \( \{\text{Emp}\} \in \max_{\Sigma, R_s}(\text{Mgr}) \) is not a proper subset of \( w(\text{Mgr}) = \{\text{Emp, Mgr}\} \). □

5.4 Characterizing the Satisfaction of FD Sets

The next theorem generalizes the result from total relations [48] saying that a total relation satisfies a set of functional dependencies precisely when the agree sets of the total relation are closed attribute sets.

Theorem 6 Let \( R \) be some relation schema, let \( \Sigma \) be an FD set, and \( R_s \) an NFS over \( R \). For all relations \( r \) over \( R \) it holds that \( r \) satisfies \( \Sigma \) and \( R_s \) if and only if \( R_s \subseteq \text{total}(r) \) and \( \forall X \in \text{ag}(r)(X_{\Sigma, R_s} \subseteq w(X)) \) holds.
Proof Sufficiency. The proof of Theorem 5 shows that \( r \) satisfies \( \Sigma \), if \( VX \in ag^r(r)(X^*_R \subseteq w(X)) \) holds. Furthermore, if \( R_s \subseteq \text{total}(r) \), then \( r \) is \( R_s \)-total.

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

Example 17 Example 16 shows that the relation from Example 2 satisfies the FD set \( \Sigma \). We specify tuples that agree sets on the maximal sets of the at-

6 Computation of Armstrong Tables

Following Section 4 we need to ask whether there is a reason to look for FDs in the presence of an NFS. 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 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 2 enables us to utilize a similar idea that Mannila and Räihä used to compute Armstrong relations for sets of FDs over total relations [75]. We specify tuples that have strong agree sets on the maximal sets of the attributes. In addition, these tuples are total and unique across 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 \( \Sigma = \Sigma' \cup \{ X \rightarrow A \} \) a set of standard functional dependencies over \( R \). For \( WC \subseteq R \), it takes \( O(|R| \times ||\Sigma||) \) time to test whether \( W \in \max_{\Sigma,R_s}(C) \).

Proof Using the NFSclosure-algorithm, \( C \notin W^*_R \) can be checked in time \( O(||\Sigma||) \), and \( C \in (W(B))^*_R \) for all \( B \in R - W \) can be checked in time \( O(|R| \times ||\Sigma||) \). □

We use \( \text{mtest}(W, C, R_s, R_s, \Sigma) \) to denote the test if \( W \in \max_{\Sigma,R_s}(C) \) from Lemma 1. The maximal sets for \( R \) with respect to \( \Sigma \) 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 \( \Sigma \) 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 \( \Sigma \) 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 \( \Sigma = \Sigma' \cup \{ X \rightarrow A \} \) a set of standard functional dependencies over \( R \). For \( C \in R \) let \( V \in \max_{\Sigma,R_s}(C) \). Then \( V \in \max_{\Sigma',R_s}(C) \)

...
Theorem 7 extends Theorem 4 in [75, 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 \((C = A \text{ or } A \in R_s)\) is always satisfied. That is, Theorem 7 becomes Theorem 4 in [75, 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 \( S \) 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 \( X \rightarrow A \in \Sigma \) do

\( \Theta := \Theta \cup \{X \rightarrow A\}; \)

for all \( C \in R \) where \((C = A \text{ or } A \in R_s)\) do

\( nnmax(C) := max(C); \)

for all \( W \in max(C) \) do

if not \( mtest(W,C,R,R_s,\Theta) \) then

\( max(C) := \{W\}; \)

for all \( B \in X \) do

if \( B \notin R_s \) then

\( max(C) := max(C) \cup \{W - B\}; \)

endif;

for all \( Z \in max(B) \) do

if \( mtest(W \cap Z,C,R,R_s,\Theta) \) then

\( max(C) := max(C) \cup \{W \cap Z\}; \)

endif;

endif;

endif;

enddo;

enddo;

enddo;

for all \( C \in R \) let \( max(C) := nnmax(C); \)

return \( max(C) \) for all \( C \in R \).

Theorem 7 shows that Algorithm 8 is correct.

Theorem 9 Algorithm 8, on input \((R,\Sigma,R_s)\), 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 \( R = KLM, \Sigma' = \{K \rightarrow L, L \rightarrow M\} \), and \( \Sigma = \Sigma' \cup \{M \rightarrow K\} \). 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 \( \Sigma' \) and \( R_s \). If \( R_s = M \) we have \( max_{\Sigma,R_s}(K) = \emptyset \) and \( max_{\Sigma,R_s}(L) = \emptyset \). This is case of Theorem 7 when a new dependency is added whose RHS is not declared null-free. In this case it is important to note that only members of \( max_{\Sigma,R_s}(K) \) 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 when a new dependency is added whose RHS is null-free. Only this special case can be regarded as similar to [75, Theorem 4].

The following algorithm computes an Armstrong table for an arbitrary set \( \Sigma \) of standard FDs and an NFS \( R_s \). 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. 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);\n
(A1) \( r := \emptyset; \)

(A2) for all \( X \in max(R) \) do

\( Z := \{A \in R \mid X \in max(A)\}; \)

(A3) \( r := r \cup \{t_i,t_{i+1}\} \) where\n
\( t_i(A) := \begin{cases} c_{A}, & \text{if } A \in XZ \cap R_s; \\ n_i \text{ otherwise, } & \end{cases} \)

\( t_{i+1}(A) := \begin{cases} c_{A}, & \text{if } A \in X; \\ c_{A,+1}, & \text{if } A \in Z(R_s - X); \\ n_i \text{ otherwise, } & \end{cases} \)

(A4) \( i := i + 2; \)

(A5) \( i := i + 1; \)

(A6) enddo;
(A7) \( total(r) := \{ A \in R \mid \forall t \in r(t[A] \neq ni) \} \)

\[
\text{if } total(r) - R_s \neq \emptyset, \text{ then return } r := r \cup \{ t_i, t_{i+1} \}
\]

\[
t_i(A) := \begin{cases} c_{A,i}, & \text{if } A = A_0 \in R \; \text{ and } \ni \; \text{, else} \\ \ni, & \text{if } A = A_0 \in R \; \text{ and } \ni \; \text{, else} 
\end{cases}
\]

\[
t_{i+1}(A) := \begin{cases} \ni, & \text{if } A = A_0 \in R \\ c_{A,i+1}, & \text{else} 
\end{cases}
\]

\[
\text{else return } r := r \cup \{ t_i \}
\]

\[
t_i(A) := \begin{cases} \ni, & \text{if } A \in total(r) - R_s \\ c_{A,i}, & \text{else} 
\end{cases}
\]

\[
\text{endif; else return } r \text{ endif}; \]

For the soundness proof of Algorithm 10 we utilize Atzeni and Morfuni’s axiomatization [6].

**Theorem 11** Algorithm 10, on input \((R, \Sigma, R_s)\), computes an Armstrong table for \(\Sigma\) 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 \(t[A] \neq ni \neq t'[A]\). 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 \(t'[A] \neq ni\), then \(t[A] \neq t'[A]\) and \(t[A] \neq ni \neq t'[A]\). If \(t[B] \neq ni\), then \(t[B] \neq t'[B]\) and \(t[B] \neq ni \neq t'[B]\). If \(t[A] = ni\) and \(t[B] = ni\), then neither of \(t, t'\) subsume 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 \(\Sigma\) and \(R_s\).

Let \(X \rightarrow A \in \Sigma\). Assume that \(r\) violates \(X \rightarrow A\). Then there are distinct \(t, t' \in r\) such that \(t[X] = t'[X]\) and \(t[A] \neq t'[A]\). Since \(X \neq 0\), 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 \(X \subseteq X' \subseteq A \in Z \subseteq R_s\). Suppose that \(A \in Z\). Consequently, \(X \subseteq X' \subseteq max(A)\), i.e., \(A \notin (X')^{Z,R_s}\). However, due to the soundness of the augmentation rule we conclude that \(A \notin X'_{Σ,R_s}\). This means that \(X \rightarrow A \notin Σ_{R_s}\), which contradicts \(X \rightarrow A \in \Sigma\). Suppose now that \(A \notin Z\). From \(X' \in max(B)\) it follows that \(X' \rightarrow B \notin Σ_{R_s}\) and that \(X' \rightarrow B \notin Σ_{R_s}\). From \(X \rightarrow A \in Σ_{R_s}\), follows \(X' \rightarrow A \in Σ_{R_s}\) by the soundness of the augmentation rule. From the soundness of the reflexivity axiom and the union rule we conclude that \(X' \rightarrow X'A \in Σ_{R_s}\). An application of the null transitivity rule to \(X' \rightarrow X'A, X'A \rightarrow B \) and \(A \in R_s\) results in \(X' \rightarrow B\). Due to the soundness of the null transitivity rule we conclude that \(X' \rightarrow B \in Σ_{R_s}\). This contradicts the fact that \(X' \in max(B)\). We have just shown that \(r\) satisfies \(\Sigma\). 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 \(X \rightarrow A \notin Σ_{R_s}\). 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 \(t[X] = t'[X]\), \(t, t' \rightarrow X\)' total and \(t[A] \neq t'[A]\). Hence, \(r\) violates \(X \rightarrow A\).

It is now quite easy to see that the relation \(r\) is an Armstrong table for \(\Sigma\) 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 \(R_s \subseteq \Sigma_{R_s}\) always holds due to the construction. Hence, if \(total(r) - R_s \neq \emptyset\), then we need to add some tuples with occurrences of \(ni\) 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 \(ni\) 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 \(R_s\)).

**Example 19** Let \(\text{EMPLOYMENT} = \{ \text{Emp, Dept, Mgr} \}\)

\(\Sigma = \{ \text{Emp} \rightarrow \text{Dept}, \text{Dept} \rightarrow \text{Mgr} \}\)

and \(R_s = \{ \text{Emp, Mgr} \}\) be as in Example 3. The following table illustrates the evolution of the maximal set families for the attributes of \(\text{EMPLOYMENT}\) using Algorithm 8 on input \(\text{EMPLOYMENT}, \Sigma, R_s\) when the first FD considered is \(\sigma = \text{Dept} \rightarrow \text{Mgr}\):

<table>
<thead>
<tr>
<th>(A)</th>
<th>(max_{\Sigma,R_s}(A))</th>
<th>(max_{\Sigma,R_s}(A))</th>
<th>(max_{\Sigma,R_s}(A))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp</td>
<td>{{\text{Dept, Mgr}}}</td>
<td>{{\text{Dept, Mgr}}}</td>
<td>{{\text{Dept, Mgr}}}</td>
</tr>
<tr>
<td>Dept</td>
<td>{{\text{Emp, Mgr}}}</td>
<td>{{\text{Emp, Mgr}}}</td>
<td>{{\text{Mgr}}}</td>
</tr>
<tr>
<td>Mgr</td>
<td>{{\text{Emp}}}</td>
<td>{{\text{Emp}}}</td>
<td></td>
</tr>
</tbody>
</table>

Based on the elements of \(max_{\Sigma,R_s}\) Algorithm 10 would compute the following Armstrong table:

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp.1</td>
<td>Dept.1</td>
<td>Mgr.1</td>
</tr>
<tr>
<td>Emp.2</td>
<td>Dept.1</td>
<td>Mgr.1</td>
</tr>
<tr>
<td>Emp.3</td>
<td>Dept.3</td>
<td>Mgr.3</td>
</tr>
<tr>
<td>Emp.4</td>
<td>Dept.4</td>
<td>Mgr.3</td>
</tr>
<tr>
<td>Emp.5</td>
<td>ni</td>
<td>Mgr.5</td>
</tr>
<tr>
<td>Emp.5</td>
<td>ni</td>
<td>Mgr.6</td>
</tr>
</tbody>
</table>

for \(\Sigma\) 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 [69]. 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 \( \Sigma \).

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 [75]. 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 [9,71].

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 \( \Sigma \) and \( R_s \) is a lower bound for the number of agree sets found in any Armstrong table for \( \Sigma \) and \( R_s \). A similar result holds for the special case of total relations [9].

Proposition 1 Let \( \Sigma \) 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 \( \Sigma \) and \( R_s \). Then \( \max_{\Sigma, R_s} (|R|) \leq |\text{ag}(r)| \leq \left( \frac{n^2}{2} \right) \).

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

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 \( \Sigma \) 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 \( \Sigma \) of standard FDs and an NFS \( R_s \) in which the number of tuples in each minimum-sized Armstrong table for \( \Sigma \) 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.

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 \( \Sigma \) of standard functional dependencies and an NFS \( R_s \) for which the number of tuples in each Armstrong table for \( \Sigma \) and \( R_s \) is exponential in the number of attributes. According to Proposition 1 it suffices to find a set \( \Sigma \) of standard FDs such that \( \max_{\Sigma, R_s} (|R|) \) is exponential in the number of attributes. Such a set \( \Sigma \) 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 functional dependencies 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 \( \Sigma \) have the same right-hand side.

\[\square\]
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 \( \Sigma \) should be of minimum size. The reason is that a small number of tuples is easier to comprehend for humans. Furthermore, if the removal of any tuple from an Armstrong table for an FD set \( \Sigma \) and an NFS \( R_s \) results in a relation that is still Armstrong for \( \Sigma \) and \( R_s \), then the tuple did not add any new information about the representation of the FD set and the NFS. For these reasons it is a practical question to ask how many tuples a minimum-sized Armstrong table requires.

An Armstrong table \( r \) for \( \Sigma \) and \( R_s \) is said to be minimum-sized if there is no Armstrong table \( r' \) for \( \Sigma \) and \( R_s \) such that \(|r'| < |r|\). That is, for a minimum-sized Armstrong table for \( \Sigma \) and \( R_s \) there is no Armstrong table for \( \Sigma \) and \( R_s \) with a smaller number of tuples.

**Proposition 3** Let \( \Sigma \) 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 \( \Sigma \) and \( R_s \). Then
\[
\sqrt{1 + 8 \cdot |\max_{\Sigma,R_s}(R)|} \leq |r| \leq 2 \times |\max_{\Sigma,R_s}(R)| + 1.
\]

**Proof** The lower bound follows from Proposition 1. Indeed, it follows that \(|\max_{\Sigma,R_s}(R)| \leq (|\Sigma|)^n\). Consequently, we have that \(\sqrt{1 + 8 \cdot |\max_{\Sigma,R_s}(R)|} \leq |r|\). 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 \(|\max_{\Sigma,R_s}(R)| + 2\) if and only if \(\text{total}(r) = R\) holds before step (A7) and \(R_s = \emptyset\). We have \(B \in \text{total}(r)\) 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 \(\text{total}(r) = R\) 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 specific case and when \(|R| > 1\), one may use the Algorithm from Mannila and Räihä [75] to compute an Armstrong relation for \(\Sigma\) 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 \(\Sigma\) and \(R_s\). If \(|R| = 1\), then the singleton relation consisting of the tuple \(t = \ni\) is an Armstrong table for \(\Sigma\) and \(R_s\).

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

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

There are also instances for which Algorithm 1 computes a minimum-sized Armstrong table.

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

**Proof** Let \(R = A_1 \cdots A_n\) with \(n \geq 3\), \(R_s = \emptyset\), and \(\Sigma\) consist of the FDs \(A_i \rightarrow A_{i+1}\) 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 \(A_1 = A_{n+1} = A_n\)). Let \(r\) be any Armstrong table for \(\Sigma\) and \(R_s\). We show that its size 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 \(ag(t_i, t'_i) = R - A_{i-1}A_i\) 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 \(A_{i-2} \rightarrow A_{i-1}\), and also \(t_i[A_i] \neq ni \neq t'_i[A_i]\) for \(i = 1, \ldots, n\) since \(r\) is subsumption-free. Hence, \(\text{total}(\{t_i\}) = R - A_{i-1} = \text{total}(\{t'_i\}) = R - A_{i-1}\) for \(i = 1, \ldots, n\). Obviously, the tuples \(t_1, t'_1, \ldots, t_n, t'_n\) are mutually distinct, so that \(r\) has size at least 2 \times \(|\max_{\Sigma,R_s}(R)|\).

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. In fact, the proof of Proposition 2 provides us with such an FD set \(\Sigma\) and the NFS \(R_s = R\).

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

The following theorem, however, 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 [72], for an FD set \(\Sigma\) and an NFS \(R_s\), we call an FD set \(\Sigma'\) an optimal cover of \(\Sigma\) with respect to \(R_s\) if

- \(\Sigma'\) is a cover of \(\Sigma\) 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 $\Sigma''$ of $\Sigma$ with respect to $R_s$ such that $\Sigma''$ contains fewer symbol occurrences than $\Sigma'$ (repeated symbol occurrences are counted as many times as they occur).

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

**Proof** Let $R = A_1B_1 \cdots A_nB_nC$, $R_s = R$ and $\Sigma = \{X_1, \ldots, X_n \rightarrow C \mid \forall i = 1, \ldots, n(X_i \in \{A_i, B_i\})\}$. We show that $\Sigma$ is the optimal cover of $\Sigma$ with respect to $R_s$, but there is an Armstrong table for $\Sigma$ and $R_s$ where the number of tuples is in $O(n)$.

We will show first that $\Sigma$ is non-redundant (no subset of $\Sigma$ implies all FDs in $\Sigma$), and then show that $\Sigma$ is an optimal cover of itself. We note that for every FD $\sigma \in \Sigma$, the closure $X^{\ast}_{(\Sigma - \{\sigma\}, R_s)}$ of $X$ with respect to $\Sigma - \{\sigma\}$ and $R_s$ is itself, i.e., $X^{\ast}_{(\Sigma - \{\sigma\}, R_s)} = X$. The reason is that there is no $\sigma' \in \Sigma - \{\sigma\}$ such that $LHS(\sigma') \sqsubseteq X$. Hence, $\Sigma$ is not implied by $\Sigma - \{\sigma\}$ and $R_s$. That is, $\Sigma$ is non-redundant.

Next we remark that every optimal cover $\Sigma'$ of $\Sigma$ with respect to $R_s$ contains only FDs $X \rightarrow Y$ such that $Y = C$. Suppose, to the contrary, that there is some FD $X \rightarrow Y$ in $\Sigma'$ such that $Y \neq C$. If $Y - X = C$ and $Y \cap X \neq \emptyset$, then $\Sigma'$ is not optimal since

$$(\Sigma' - \{X \rightarrow Y\}) \cup \{X \rightarrow Y - X\}$$

is equivalent to $\Sigma$ but contains less attributes than $\Sigma'$. If $Y - X = \emptyset$, then $\Sigma' - \{X \rightarrow Y\}$ is equivalent to $\Sigma$ but contains less symbol occurrences than $\Sigma'$. If $Y - X \neq \emptyset$ and $Y - X \neq C$, then $\Sigma \not\models_{R_s} X \rightarrow Y$ and, therefore, $\Sigma'$ is not a cover of $\Sigma$ with respect to $R_s$. Moreover, every FD $X \rightarrow Y$ in an optimal cover $\Sigma'$ of $\Sigma$ with respect to $R_s$ satisfies that $C \notin X$. If there was an FD $X \rightarrow C \in \Sigma'$ and $C \in X$, then

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

is equivalent to $\Sigma$ but contains less attributes than $\Sigma'$.

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

Thus we have just shown that $\Sigma$ 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 $\Sigma$ and $R_s$ where the number of tuples is in $O(n)$. 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 $\max_{\Sigma,R_s}(R)$ 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 Mamala and Rāhīa’s algorithm [75] (which applies since we are in the case where $R_s = R$) we can easily create an Armstrong table for $\Sigma$ and $R_s$ that has $3n + 1$ tuples only.

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, 71] 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 $\Sigma$ 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 $\Sigma$ 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. The hardness part of the next result follows from the fact that we can 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 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 \( \Sigma \) of standard FDs and an NFS \( R_s \) in the world of all FDs, if \( \text{total}(r) = R_s \) and the following holds for all standard and non-standard FDs \( \sigma \): \( r \) satisfies \( \sigma \) if and only if \( \Sigma \vdash_{R_s} \sigma \). 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 \( \Sigma \) of standard FDs and an NFS \( R_s \) in the world of all FDs, if \( \text{total}(r) = R_s \), \( r \) violates every non-trivial non-standard FD, and the following holds for all standard FDs \( \sigma \): \( r \) satisfies \( \sigma \) if and only if \( \Sigma \vdash_{R_s} \sigma \).

Every Armstrong table \( r \) for a set \( \Sigma \) of standard FDs and an NFS \( R_s \) can be easily modified to obtain an Armstrong table for \( \Sigma \) and \( R_s \) in the world of all FDs: just add to \( r \) an additional tuple with previously unused domain values different from \( \pi \). 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 \( \emptyset \rightarrow A \). Consequently, no Armstrong table exists for the empty FD set \( \Sigma \) 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 \( t = \pi \) is an Armstrong table for the empty FD set \( \Sigma \) and the NFS \( R_s = \emptyset \) in the “world of all 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.

It is now a natural question to ask how 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 1 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 20** Consider \( R = \{ \text{Dept}, \text{Mgr} \} \) with standard FD set \( \Sigma = \{ \text{Dept} \rightarrow \text{Mgr} \} \) and NFS \( R_s = R \). The relation \( r \) is an Armstrong table for \( \Sigma \) and \( R_s \), but not an Armstrong table for \( \Sigma \) and \( R_s \) in the world of all FDs: the non-standard FD \( \emptyset \rightarrow \text{Mgr} \) is not implied by \( \Sigma \) 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 \( \text{max}_{\Sigma, R_s} (\text{Mgr}) - \alpha^*(r) \).

If we add the tuple \( t = (\text{CS}, \text{von Neumann}) \) to \( r \), then \( r \cup \{t\} \) is an Armstrong table for \( \Sigma \) and \( R_s \) in the world of all FDs.

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 Section 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. We analyze a detailed example in which the provision of an Armstrong table can result in database designs that facilitate efficient updates and queries, and are less prone to inference attacks.

9.1 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.

Suppose that the designers of our information system have identified the following set \( \Sigma \) of FDs to be meaningful:

<table>
<thead>
<tr>
<th>Dept</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>Gauss</td>
</tr>
<tr>
<td>Physics</td>
<td>Gauss</td>
</tr>
</tbody>
</table>
Furthermore, the design team has agreed to declare the attributes \( \text{Order}_s \) and \( \text{Product}_s \) as NOT NULL, i.e.,

\[
\text{ORDER}_s = \{ \text{Order}_s, \text{Product}_s \}.
\]

Before the team goes ahead with the implementation they decide to validate their design using an Armstrong table for \( \Sigma \) and \( \text{ORDER}_s \). Using Algorithm 8 the family of maximal sets can be computed as in Table 2. Furthermore, Table 3 shows an Armstrong table for \( \Sigma \) and \( \text{ORDER}_s \). The design team observes from the table that the product with \( \text{Product}_s \) 612 occurs with the same total value \( \text{Total} \) of 25000, but with the different quantities \( \text{Qty} \) of 75 and 100. After consultation with the domain experts, the design team responds by including the additional meaningful FD

\[
\text{Product}_s, \text{Total} \rightarrow \text{Qty}.
\]

in \( \Sigma \). By further inspecting this table, the design team notices that \( \text{Codd} \{ \{ \text{Order}_s, \text{Product}_s \} \} \) is not implied by the current design choice, in contrast to the intention of the team. In particular, the product with \( \text{Product}_s \) 834 in the order with \( \text{Order}_s \) 43056 has different total values of 35000 and 40000. The design team has at least two choices to overcome the shortcomings of their current design. As a first choice they could leave \( \Sigma \) unchanged, but specify the attribute \( \text{Qty} \) as NOT NULL (i.e. \( \text{ORDER}_s = \{ \text{Order}_s, \text{Product}_s, \text{Qty} \} \)). As a second choice they could leave \( \text{ORDER}_s \) unchanged, but specify the additional functional dependency

\[
\text{Order}_s, \text{Product}_s \rightarrow \text{Total}.
\]

After consultation with the domain experts, the design team decides to go ahead with the first choice. In fact, \( \Sigma \) and \( \text{ORDER}_s = \{ \text{Order}_s, \text{Product}_s, \text{Qty} \} \) together imply the FD \( \text{Order}_s, \text{Product}_s \rightarrow \text{Total} \). Note that the previous design did not successfully capture neither the meaningful FD \( \text{Product}_s, \text{Total} \rightarrow \text{Qty} \) nor the meaningful Codd key \( \text{Codd} \{ \{ \text{Order}_s, \text{Product}_s \} \} \). Consequently, the inspection of the Armstrong table has resulted in the recognition of additional meaningful business rules for the application domain. We will now illustrate the impact of having correctly captured these meaningful constraints on various data processing tasks.

9.2 Efficient Processing of Updates

The main driver of database normalization is to avoid data redundancy in any of the future database instances. The reason is that data redundancy can lead to inefficiencies whenever the database instance is subject to updates. The Boyce-Codd Normal Form (BCNF) enforces a syntactic condition on relation schemata that guarantees the absence of data redundancies in terms of FDs [8, 12, 87]. Hence, schemata in BCNF are also free from update anomalies and enjoy efficient constraint management [8, 12, 87]. We assume familiarity with the definition of such terms as BCNF, lossless and dependency-preserving decomposition [1].

Our example relation schema \( \text{ORDER} \) is not in BCNF with respect to the FD set \( \Sigma \) containing

\[
\begin{align*}
\text{Order}_s, \text{Product}_s \rightarrow \text{Qty}, \\
\text{Product}_s \rightarrow \text{Description}, \\
\text{Product}_s, \text{Qty} \rightarrow \text{Total}, \text{and} \\
\text{Product}_s, \text{Total} \rightarrow \text{Qty},
\end{align*}
\]

and the NFS \( \text{ORDER}_s = \{ \text{Order}_s, \text{Product}_s, \text{Qty} \} \). Following the BCNF decomposition strategy [1], we can decompose the schema \( \text{ORDER} \) into the schema \( \text{PRODUCT} = \{ \text{Product}_s, \text{Description} \} \) with FD set

\[
\Sigma_1 = \{ \text{Product}_s \rightarrow \text{Description} \}
\]

and NFS \( \text{PRODUCT}_s = \{ \text{Product}_s \} \), the schema \( \text{TOTAL} = \{ \text{Product}_s, \text{Qty}, \text{Total} \} \) with FD set \( \Sigma_2 \) containing

\[
\begin{align*}
\text{Product}_s, \text{Qty} \rightarrow \text{Total} \text{ and Product}_s, \text{Total} \rightarrow \text{Qty}, \text{ and} \\
\text{NFS TOTAL}_s = \{ \text{Product}_s, \text{Qty} \}, \text{ and the schema} \\
\text{QTY} = \{ \text{Order}_s, \text{Product}_s, \text{Qty} \} \text{ with FD set} \\
\Sigma_3 = \{ \text{Order}_s, \text{Product}_s \rightarrow \text{Qty} \},
\end{align*}
\]

and NFS \( \text{QTY}_s = \{ \text{Order}_s, \text{Product}_s, \text{Qty} \} \). We can easily verify that the three relation schemata

\[
\begin{align*}
\text{(PRODUCT,}\Sigma_1,\text{PRODUCT}_s)
\end{align*}
\]

\[
\begin{align*}
\text{ORDER}_s & \rightarrow \{ \{ \text{Product}, \text{Description}, \text{Qty}, \text{Total} \} \} \\
\text{Product}_s & \rightarrow \{ \{ \text{Order}, \text{Description}, \text{Qty}, \text{Total} \} \} \\
\text{Description} & \rightarrow \{ \{ \text{Order}, \text{Description}, \text{Qty}, \text{Total} \} \} \\
\text{Qty} & \rightarrow \{ \{ \text{Order}, \text{Description}, \text{Qty}, \text{Total} \} \} \\
\text{Total} & \rightarrow \{ \{ \text{Order}, \text{Product}, \text{Description} \} \} \\
\end{align*}
\]

\[
\begin{align*}
\text{Table 2} \text{ Maximal set families for } \Sigma \text{ and } \text{ORDER}_s
\end{align*}
\]

\[
\begin{align*}
\text{Table 3} \text{ Armstrong table for } \Sigma \text{ and } \text{ORDER}_s
\end{align*}
\]
- \((\text{Total}, \Sigma_2, \text{Total}_a)\)
- \((\text{Qty}, \Sigma_3, \text{Qty}_a)\)

represent a lossless and dependency-preserving BCNF decomposition of \text{ORDER} for \Sigma and \text{ORDER}_a. Note that for the schema \text{ORDER} with the original FD set \Sigma and NFS \text{ORDER}_a = \{\text{Order}_a, \text{Product}_a\} every BCNF decomposition is lossy or not dependency-preserving. Indeed, if we decompose using the FD \text{Product}_a, \text{Qty} \rightarrow \text{Total}, then the decomposition is lossy since \text{Qty} does not belong to \text{ORDER}_a, and if we decompose using the FD \text{Order}_a, \text{Product}_a \rightarrow \text{Qty}, then we cannot preserve the FD \text{Product}_a, \text{Qty} \rightarrow \text{Total}.

In summary, an inspection of the Armstrong table has enabled the design team to find a database layout that represents all of the meaningful business rules, permits efficient consistency checking, is free from data redundancies and update anomalies.

9.3 Efficient Processing of Queries

Besides updates, the efficient processing of database queries is also a significant task of database management systems. We will illustrate now 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 design team to identify the meaningful FD \text{Product}_a, \text{Total} \rightarrow \text{Qty} and to specify \text{Qty} as \text{NOT NULL}. In particular, the Codd key \text{Codd}(\{\text{Order}_a, \text{Product}_a\}) is implied by \Sigma and \text{ORDER}_a.

Consider first 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 would be

```
SELECT \text{ORDER}'\_a.\text{Order}_a, \text{ORDER}'\_a.\text{Qty}
FROM \text{ORDER}, \text{ORDER}\_a AS \text{ORDER}'
WHERE \text{ORDER}'\_a.\text{Product}_a=\text{ORDER}'\_a.\text{Product}_a AND
\text{ORDER}'\_a.\text{Total}_a=\text{ORDER}'\_a.\text{Total}_a
```

However, since \text{Product}_a, \text{Total} \rightarrow \text{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 \text{ORDER}.\text{Order}_a, \text{ORDER}.\text{Qty}
FROM \text{ORDER}
WHERE \text{ORDER}.\text{Qty} \geq 100
```

However, since \Sigma and \text{ORDER}_a imply the Codd key \text{Codd}(\{\text{Order}_a, \text{Product}_a\}) the \text{DISTINCT} clause is superfluous. As duplicate elimination often requires an expensive sort of the query result [79], the results we develop in this article can help to identify automatically unnecessary \text{DISTINCT} clauses.

9.4 Inference Control

Inference control is a security mechanism developed to ensure confidentiality in databases [15,16,46]. The objective is to avoid inferences of secrets by users based on their query history and their knowledge about the database. Suppose, for example, the fact that there is an order of a \textit{Fridge} with a total value of 100000 is a business secret that some users are not allowed to learn. That is, the fact that the sentence \(\exists X_O \exists X_P \exists X_Q \text{ORDER}(X_O, X_P, \text{Fridge}, X_Q, 100000)\) is true in the database must not be revealed to unauthorized users. A user may issue the queries:

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

and learn that both queries are true in the current instance, since neither \Phi_1 nor \Phi_2 individually reveal \Psi. The user may anticipate that the instance satisfies \text{Order}_a, \text{Product}_a \rightarrow \text{Description}, \text{Total}, based on the semantics of the application domain. Hence, the user may apply this FD to the answers to \Phi_1 and \Phi_2 to infer that \(\exists X_Q \text{ORDER}(73956, 1645, \text{Fridge}, X_Q, 100000)\) is true in the database instance. This, however, would reveal the potential secret to the user. Fortunately, the design team was able to utilize Armstrong tables to recognize that the FD \text{Order}_a, \text{Product}_a \rightarrow \text{Description}, \text{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 \Phi_2 after the user has learned \Phi_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.

10 Conclusion and Future Work

We have investigated the existence and properties of Armstrong relations 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 relations, in general. However, the class of standard FDs and NFSs does enjoy Armstrong relations. We have given sufficient and necessary conditions for a given relation to be an Armstrong relation for a given set of standard FDs and a given NFS. In general, the problem of finding an Armstrong relation remains precisely exponential in the number of attributes. However, we have established a provably-correct algorithm that computes an Armstrong relation in time that is at most quadratic in the size of a minimum-sized Armstrong relation. We have demonstrated that database designers should utilize not only an abstract representation in form of FD sets, but also the representation in form of an Armstrong relation. Finally, we have illustrated that the utilization of Armstrong relations can lead to the recognition 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 overheads when compared to the previously studied case of total relations.

Fagin has shown that Armstrong relations exist for so-called implicational dependencies [41]. 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 [57, 69, 70].

A main observation for the impact areas of Section 9 is that most of the existing theory does not apply to SQL tables. These areas include normalization [39], semantic query optimization [20, 35], consistent query answering [21] and controlled query evaluation [13].

We plan to implement our concepts and algorithms to extend design aids available for total relations [27, 28, 75, 81]. It seems intuitive that design teams find it more difficult to understand the interaction of FDs in the presence of an NFS than in the case of total relations. Hence, Armstrong tables might be of even bigger value than reported for the case of total relations [64].

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.

Recently, 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.

Other directions include the problem of dependency inference [76] or data cleaning [45], but also the investigation of extremal problems [32, 38] in the context of partial relations. Finally, all these problems should also be investigated for other interpretations of null values [24, 51–53, 74].

References


52. Grahne, G. Dependency satisfaction in databases with incomplete information. In *Proceedings of the Tenth International Conference on Very Large Data Bases (VLDB)* (1984), Morgan Kaufmann, pp. 37–45.


