CDMTCS
Research
Report
Series

Automated Reasoning about the Entity Integrity of Big Data in Possibilistic SQL

Ilya Litvinenko
The University of Auckland
Ziheng Wei
The University of Auckland
Sebastian Link
The University of Auckland

CDMTCS-541
January 2020

Centre for Discrete Mathematics and
Theoretical Computer Science

Automated Reasoning about the Entity Integrity of Big Data in Possibilistic SQL

Ilya Litvinenko
The University of Auckland, New Zealand
ilit874@aucklanduni.ac.nz

Ziheng Wei
The University of Auckland, New Zealand
z.wei@auckland.ac.nz

Sebastian Link
The University of Auckland, New Zealand
s.link@auckland.ac.nz

January 23, 2020

Abstract

SQL is the de-facto industry standard for data management. Besides relational data, SQL has been extended to also manage object-relational and Web-based data. It is likely that many instances of big data will also be managed by extensions of the current SQL standard. We introduce the classes of keys and functional dependencies over possibilistic databases with duplicate and missing information. Our main contribution is to equip SQL with reasoning capabilities about the semantics of big data that may feature the volume, variety, and veracity dimensions. These capabilities are fundamental to reason about entity integrity and essential for database design as functional dependencies are sources of data redundancy, and keys prevent data redundancy. Since SQL controls the occurrences of missing information with `NOT NULL` constraints, we also include possibilistic extensions of this constraint in our investigation. We illustrate applications, and establish axiomatic, algorithmic, and logical characterizations to the PTIME-complete implication problem associated with the combined class of these integrity constraints. Specifically, we show that keys behave just like goal clauses and FDs just like definite clauses in Boolean propositional Horn logic, and we can therefore apply linear resolution to reason about them.

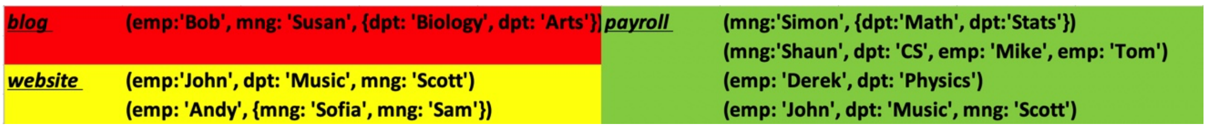
Keywords: Axiomatization; Big data; Duplicates; Functional dependency; Horn clause; Implication problem; Key; Missing values; Null markers; Possibility theory; SQL;

1 Introduction

Database management systems model some domain of the real-world within a database system. For that purpose, they use a specific structure in which the data is organized, but also integrity constraints that restrict the database instances to those that are considered meaningful for the underlying domain [23]. SQL has been the de-facto industry standard for the management of data since the 1980s. Originally introduced by Codd for application domains where data can be governed by the rigid structure of relations [5], modern applications demand more flexibillity. Big data is often characterized by at least three dimensions, including data volume, variety, and veracity. It means that large volumes of data must be handled that originate from heterogeneous sources (variety) with different degrees of uncertainty (veracity) [25]. Given the mature and popular technology that SQL provides it is not surprising that many organizations will also use SQL to manage big data, at least in one form or another. Of course, other data models will continue to arise for handling big data but history for object-relational and Web-based data have shown that SQL will evolve (to NewSQL) to take the best out of both worlds [20].

In this article we will introduce the class of keys and functional dependencies over possibilistic SQL data, with the aim to provide capabilities to efficiently reason about the entity integrity of big data that accommodates the volume, variety and veracity dimensions. Codd, the inventor of the relational model, stipulated entity integrity as one of the three major integrity principles in databases [5],. Here, entity integrity refers to the principle of representing each entity of the application domain uniquely within the database. Violations of this (and other) principle(s) are very common in database practice, resulting in their own fields of research including entity resolution [4], data de-duplication [19], data cleaning [10], or consistent query answering [3]. Keys and functional dependencies (FDs) are fundamental to entity integrity as they stipulate uniqueness of rows on a given set of columns, or uniquely determine the values on some columns given the values on some other columns. Both classes are also essential to logical and physical database design since FDs may cause data redundancy, while keys prevent them, and therefore balance the trade-off between efficient queries (based on redundant data values) and efficient updates (based on the absence of redundant values). As a consequence, efficient reasoning about keys and FDs is the foundation for effective data management [23].

While concepts such as keys and FDs have unique and unchallenged definitions in the relational model of data, simple extensions of the relational model introduce opportunities to define these concepts differently to accommodate different applications [23]. SQL, for example, provides simple means to accommodate missing information. In fact, it permits occurrences of a so-called *null marker*, denoted by $\perp$, to say that there is no information about the value of this row on this column [12, 26]. Moreover, columns can be defined to be `NOT NULL`, which means that no occurrences of the null marker are permitted to occur in such columns. The interpretation of $\perp$ is deliberately kept as simple as possible to uniformly accommodate many types of missing information, including values that do not exist or values that exist but are currently unknown [26]. While a distinction of such information is possible in principle, it would lead to an application logic that has been identified as being too complex to accommodate in database practice [6]. In modern applications, for example data integration, null marker occurrences are heavily



blog	(emp:'Bob', mng: 'Susan', {dpt: 'Biology', dpt: 'Arts'})	*payroll*	(mng:'Simon', {dpt:'Math', dpt:'Stats'})
			(mng:'Shaun', dpt: 'CS', emp: 'Mike', emp: 'Tom')
website	(emp:'John', dpt: 'Music', mng: 'Scott')		(emp: 'Derek', dpt: 'Physics')
	(emp: 'Andy', {mng: 'Sofia', mng: 'Sam'})		(emp: 'John', dpt: 'Music', mng: 'Scott')

Figure 1: JSON data from different information sources

used by SQL to fit data - originating from heterogeneous sources - within a uniform table. This is essentially SQL's answer to the variety dimension of big data. Another interesting difference between SQL and the relational model is the duplication of rows. While the relational model has set semantics, SQL has based on a multiset semantics. If T denotes a given set of attributes, the key $u(X)$ is satisfied by a set of rows if and only if the FD $X \to T$ is satisfied by the set of rows. This is no longer true for multisets of rows as the multiset of two rows with matching values on all the columns in T will violate any key and satisfy any FD [14]. Hence, for SQL we need to study the combined class of keys and FDs. The idea behind the veracity dimension is that not all data are equal and should therefore not have the same weight in data analysis and decision making. Probabilistic databases have been investigated in depth over recent years and have led to various different probabilistic data models [22]. An orthogonal approach are so-called possibilistic databases. For a detailed comparison we refer the reader to [9, 21], but remark that probabilistic databases offer continouous degrees of uncertainty but real probability distributions are hard to come by and maintain, while possibilistic databases offer discrete degrees of uncertainty and are simpler to come by and maintain.

Contributions and Organization. After introducing our running example in Section 2, we define our possibilistic SQL data model in Section 3. As our first main contribution we introduce the classes of keys, FDs, and `NOT NULL` constraints in possibilistic SQL in Section 4. The implication problem and its centrality for data management are explained in Section 5. In Sections 6 and 7, respectively, we establish axiomatic and linear-time algorithmic characterizations for the associated implication problem. As a logical characterization, we show in Section 8 that the implication problem of our combined class is equivalent to propositional HornSAT, showing that the implication problem is PTIME-complete but also solvable with linear resolution. Hence, we can reason about the entity integrity of big data very efficiently and with the help of very familiar tools. We conclude in Section 9. Note that our presentation focuses on the main results and their illustration on examples. We also include the main ideas for the proofs of our results in the main body, but the full proofs and additional results are only made available in the appendix.

2 The Running Example

As a simple running example consider Figure 1 that shows some JSON data, harvested from different information sources. JSON is the de-facto standard for data exchange due to its capability to accommodate different information structures [18]. In our example, the information origins from three sources: payroll data, web data, and blog data. As the data stewards associate different levels of trust with these sources, they would like to attribute these levels of trust to the data elements originating from the sources. Our company is using an SQL-based database management system, and the data stewards have transformed the JSON data into SQL-compliant format as shown in Table 1.

Table 1: A University Employment Table

row	*emp*	*dpt*	*mng*	p-degree	interpretation	origin
1	$\perp$	Math	Simon	α_1	fully possible	payroll
2	$\perp$	Stats	Simon	α_1	fully possible	payroll
3	Mike	CS	Shaun	α_1	fully possible	payroll
4	Tom	CS	Shaun	α_1	fully possible	payroll
5	Derek	Physics	$\perp$	α_1	fully possible	payroll
6	John	Music	Scott	α_1	fully possible	payroll
7	John	Music	Scott	α_2	quite possible	website
8	Andy	$\perp$	Sofia	α_2	quite possible	website
9	Andy	$\perp$	Sam	α_2	quite possible	website
10	Bob	Biology	Susan	α_3	somewhat possible	blog
11	Bob	Arts	Susan	α_3	somewhat possible	blog

In particular, the null marker $\perp$ has been used whenever no information was available for a data element for a given attribute. For instance, no information is available for any employee working in the Math or Stats departments. Moreover, an extra column *p-degree* has been introduced to accommodate the levels of trust associated with the data elements. Here, the highest p-degree α_1 is assigned to data from the payroll, α_2 to web data, and α_3 to blog data. We will use this application scenario to illustrate our concepts and results throughout the article.

3 Possibilistic SQL Tables

This section summarizes basic notions from [12] to introduce SQL table schemata and tables, before generalizing them to possibilistic ones.

An *SQL table schema*, denoted by T, is a finite non-empty set of attributes. Each attribute $A \in T$ has a domain $dom(A)$, which is a countably infinite set of values. Each domain contains the null marker as a distinguished element, which we denote by $\perp$. As a running example, we use the table schema WORK=$\{emp, dpt, mng\}$ which collects information about employees that work in departments under managers. The domain of each attribute is `STRING`. Next we assign a semantics to table schemata, which tells us what database instances we consider.

A *row* (or *tuple*) over table schema T is a function $r : T \to \cup_{A\in T} dom(A)$ that assigns to every attribute a value from the attribute's domain. The image $r(A)$ of a row r on attribute A is called the *value* of r on A. For $X \subseteq T$, we say that a row r over T is X-*total*, if $r(A) \neq\perp$ for all $A \in X$. We say that row r is total, if it is T-total.

Note that rows accommodate incomplete information: the "value" of a row may be $\perp$. Throughout the thesis we adopt Zaniolo's interpretation of the null marker $\perp$ as "no information" [26]. More concisely, the occurrence $r(A) =\perp$ is interpreted as having available no information about the value of row r on attribute A. In particular, it may indicate that there is no value at all, or that there is a value which is currently unknown, but we do not know which interpretation holds. Note that SQL adopts this interpretation. The interpretation has been used in several previous papers on incomplete databases, including [13, 15, 26].

Besides partiality, our data model also accommodates duplicate rows as we define tables to be multisets of rows. An *SQL table* over table schema T is a finite multiset t of rows over T. For $X \subseteq T$ we say that the table t is X-total, if every row $r \in t$ is X-total. We say that t is total, if t is T-total. A total table is sometimes called a bag. Table 2 shows an example of a table over table schema WORK. The third row has value *Derek* on *emp*, value *Physics* on *dpt*, and marker $\perp$ on *mng*. The first and second row are total, and the third and fourth row, as well as the table itself are $\{emp, dpt\}$-total. The first and second row, as well as the third and fourth row, respectively, constitute duplicate tuples, since they have matching values on all attributes.

Table 2: Table over WORK

emp	*dpt*	*mng*
John	Music	Scott
John	Music	Scott
Derek	Physics	$\perp$
Derek	Physics	$\perp$

Relational databases and SQL tables were developed to manage data that is fully certain. As a consequence, there is no mechanism to distinguish between different degrees of uncertainty. For example, all tuples of a table occur with full certainty and one cannot say that tuple (Derek, Physics, $\perp$) is less likely to occur than tuple (John, Music, Scott).

We will now extend the data model by assigning degrees of possibilities (p-degrees) to tuples, also extending the possibilistic data model introduced in [16, 17] over the pure relational model of data. In our running example the p-degrees result from the source the tuples originate from. In fact, the tuples in Table 1 about university employees have been integrated from payroll data, data found on websites, and blogs. In the application, payroll data is considered to be 'fully possible', website data to be 'quite possible', and blog data to be 'somewhat possible', while other potential tuples not in the database instance are viewed to be 'impossible' to occur at the moment. In general, p-degrees can have different interpretations and origins, and we will simply denote them by abstract symbols $\alpha_1, \ldots, \alpha_k, \alpha_{k+1}$. Here, p-degrees form a finite linear order. Table 1 shows an instance in which tuples have been assigned p-degrees.

The table includes meta-data: the column 'row' assigns a unique identifier to each tuple, and the columns 'interpretation' and 'origin' show the interpretation of the p-degrees, as well as the source where the tuples originate from.

A *possibility scale* is a strict finite linear order $\mathcal{S}_p = (S_p, >_p)$, denoted by $\alpha_1 >_p \cdots > \alpha_k >_p \alpha_{k+1}$, where k is at least one. The elements α_i of S_p are called *possibility degrees* (p-degrees). The top degree α_1 is reserved for rows which are 'fully possible', while the bottom degree α_{k+1} is reserved for rows which are 'impossible'.

In Table 1 of our running example we have $k = 3$ for the underlying *possibility scale.* The fully possible rows appear at the top with the p-degree α_1, while the least possible ones appear at the bottom with the p-degree α_3. The scale also includes the p-degree $\alpha_{k+1} = \alpha_4$. However, this degree corresponds to 'impossible' rows, or all the rows which could potentially be constructed from the corresponding domains, but are not included in Table 1. SQL tables with such a possibility scale give us additional information about the rows enabling us to make use of the uncertainty associated with the scale. It should be noted that non-possibilistic tables form the special case of possibilistic ones where $k = 1$. These comments and our example motivate the following definitions.

A *possibilistic SQL table schema* (or p-SQL table schema) is a pair $(T, \mathcal{S}_p)$, where T is a table schema and $\mathcal{S}_p$ is a possibility scale. Possibilistic SQL tables are SQL tables in which each row is assigned a non-bottom p-degree. Rows that do not belong to the table are associated with the bottom p-degree. A *possibilistic SQL table* (or p-SQL table) over the possibilistic table schema $(T, \mathcal{S}_p)$ consists of a table t over T, and a function $Poss_t$ that maps each row $r \in t$ to a p-degree $Poss_t(r) \neq \alpha_{k+1}$ in the p-scale $\mathcal{S}_p$. The p-SQL table of our running example is shown in Table 1. It consists of an SQL table over WORK in which every row is assigned a p-degree from α_1, α_2 or α_3. P-SQL tables enjoy a well-founded possible world semantics. Indeed, the possible worlds form a linear chain of k SQL tables in which the i-th possible world contains tuples with p-degree α_i or higher. Given a possibilistic SQL table t over the possibilistic SQL table schema $(T, \mathcal{S}_p)$, the possible world t_i associated with t is defined by $t_i = \{r \in t \mid Poss_t(r) \geq \alpha_i\}$, that is, t_i is an SQL table of those rows in t that have a p-degree α_i or higher.

It should be noted that t_{k+1} is not considered to be a possible world as it would contain impossible tuples. Continuing our running example, Table 3 shows the possible worlds of the possibilistic SQL table from Table 1. Indeed, the possible worlds of t form a linear chain $t_1 \subseteq t_2 \subseteq t_3$. While the chain is strict here, this does not need to be the case in general as there may be non-bottom p-degrees which are not assigned to any tuples.

Table 3: Possible Worlds for the Possibilistic SQL Table from Table 1

t_1

emp	*dpt*	*mng*
$\perp$	Math	Simon
$\perp$	Stats	Simon
Mike	CS	Shaun
Tom	CS	Shaun
Derek	Physics	$\perp$
John	Music	Scott

t_2

emp	*dpt*	*mng*
$\perp$	Math	Simon
$\perp$	Stats	Simon
Mike	CS	Shaun
Tom	CS	Shaun
Derek	Physics	$\perp$
John	Music	Scott
John	Music	Scott
Andy	$\perp$	Sofia
Andy	$\perp$	Sam

t_3

emp	*dpt*	*mng*
$\perp$	Math	Simon
$\perp$	Stats	Simon
Mike	CS	Shaun
Tom	CS	Shaun
Derek	Physics	$\perp$
John	Music	Scott
John	Music	Scott
Andy	$\perp$	Sofia
Andy	$\perp$	Sam
Bob	Biology	Susan
Bob	Arts	Susan

The linear order of the p-degrees $\alpha_1 > \cdots > \alpha_k$ results in a reversed linear order of possible worlds associated with a p-SQL table t: $t_1 \subseteq \cdots \subseteq t_k$. We point out the distinguished role of the top p-degree α_1. Every row that is fully possible belongs to every possible world. Therefore, every fully possible row is also fully certain. This explains why p-SQL tables subsume SQL tables as a special case.

4 Possibilistic SQL Constraints

We introduce possibilistic keys, FDs, and `NOT NULL` constraints to restrict SQL tables to those that actually occur in an application domain.

SQL Constraints. We recall the definitions of SQL functional dependencies and `NOT NULL` constraints [12, 13]. Keys have previously not been investigated in this context, but are essential to entity integrity and cannot be expressed by FDs, in contrast to the special case where multisets of partial rows reduce to sets of complete rows.

Intuitively, a key is a collection of attributes which can separate different rows by their values on the key attributes. The following definition adopts the semantics for the SQL constraint `UNIQUE` and separates different rows whenever they are total on all the key attributes. A *key* over an SQL table schema T is an expression $u(X)$ where $X \subseteq T$. An SQL table t over T is said to *satisfy* the key $u(X)$ over T, denoted by $\models_t u(X)$ if for all $r_1, r_2 \in t$ the following holds: if $r_1(X) = r_2(X)$ and r_1, r_2 are X-total, then $r_1 = r_2$. For example, the possible world t_1 of Table 3 satisfies $u(emp)$, while t_2 and t_3 violate this key.

The following semantics of FDs goes back to Lien [15]. A *functional dependency* (FD) over an SQL table schema T is an expression $X \to Y$ where $XY \subseteq T$. An SQL table t over T is said to *satisfy* the FD $X \to Y$ over T, denoted by $\models_t X \to Y$ if for all $r_1, r_2 \in t$ the following holds: if $r_1(X) = r_2(X)$ and r_1, r_2 are X-total, then $r_1(Y) = r_2(Y)$. For example, the possible world t_2 of Table 3 satisfies the FDs *emp* $\to$ *dpt* and *dpt* $\to$ *mng*, while t_3 satisfies the FD *dpt* $\to$ *mng*, but not the FD *emp* $\to$ *dpt*.

SQL `NOT NULL` constraints offer a convenient mechanism to control occurrences of the null marker. They have been studied in combination with FDs and also multivalued dependencies [13]. A `NOT NULL` constraint over an SQL table schema T is an expression $n(X)$ where $X \subseteq T$. An SQL table t over T is said to *satisfy* the `NOT NULL` constraint $n(X)$ over T, denoted by $\models_t n(X)$, if t is X-total. For a given set Σ of constraints over T we call $T_s = \{A \in T \mid \exists n(X) \in \Sigma \wedge A \in X\}$ the *null-free subschema* (NFS) over T. If $T_s = T$, we also call T a *bag schema*, because all instances over T are bags. For example, $n(dpt)$ is satisfied by the possible world t_1 in Table 3, but not by t_2 or t_3.

Possibilistic SQL Constraints. We will now extend our semantics of SQL constraints to possibilistic SQL tables. Following the same approach as in [16], we use the p-degrees of rows to specify with which certainty an SQL constraint holds. Similar to how α_i denotes p-degrees of rows, β_i denotes c-degrees by which constraints hold. Before introducing the formal semantics, we will look at some constraints in the context of our running example. Let us consider the following SQL constraints: *dpt* $\to$ *mng*, *emp* $\to$ *dpt*, *u(emp)*, and *n(emp)*, and inspect the possible worlds t_1, t_2, t_3 in Table 3.

- *dpt* $\to$ *mng* : The constraint is satisfied by the largest possible world t_3, and therefore in t_2 and t_1 as well. Since the constraint is satisfied by every possible world, we can say that it is 'fully certain' to hold, denoted by β_1.
- *emp* $\to$ *dpt* : The constraint is satisfied by the second largest possible world t_2 and therefore by t_1, but it is not satisfied by t_3. Since the constraint is only violated by the 'somewhat possible' world t_3, we can say that it is 'quite certain' to hold, denoted by β_2.

- *u(emp)* : The constraint is satisfied by the smallest possible world t_1, but it is not satisfied by t_2 and therefore not by t_3. Since the smallest possible world that violates the constraint is 'quite possible', we can say that it is 'somewhat certain' to hold, denoted by β_3.

- *n(emp)* : The constraint is not even satisfied in the 'fully possible' world t_1. We can therefore say that it is 'not certain at all' to hold, denoted by β_4.

We can see from the examples above how the p-degrees of rows result in degrees of certainty (c-degrees) with which constraints hold on p-SQL tables. Essentially, if the smallest world that violates a constraint has p-degree α_i (this world is impossible only when all possible worlds satisfy the constraint), then the constraint holds with c-degree β_{k+2-i}. For example, the p-key *u(emp)* holds with c-degree β_3 in the p-SQL table t of Table 1, meaning that the smallest possible world that violates *u(emp)* is t_2 which is 'quite possible', or in other words, *u(emp)* is 'somewhat certain' to hold in t. The following definition introduces the certainty scale that is derived from a given possibility scale.

Let $(T, \mathcal{S}_p)$ denote a p-SQL table schema where the bottom p-degree of $\mathcal{S}_p$ is $k+1$. The certainty scale $\mathcal{S}_p^T$ associated with $(T, \mathcal{S}_p)$ is the strict finite linear order $\beta_1 >_p \cdots >_p \beta_k >_p \beta_{k+1}$. The top c-degree β_1 is reserved for constraints that are 'fully certain', while the bottom c-degree β_{k+1} is reserved for constraints that are 'not certain at all'.

Next we define by which c-degree a given SQL constraint holds on a given p-SQL table. Similar to the concept of a marginal probability in probability theory, we call this c-degree the marginal certainty. In traditional SQL tables an SQL constraint either holds or does not hold. In a p-SQL table, an SQL constraint always holds with some c-degree.

Definition 1 (Marginal certainty) *Let σ denote an SQL key, FD or* `NOT NULL` *constraint over table schema T. The* marginal certainty $c_t(\sigma)$ *by which σ holds in the p-SQL table t over $(T, \mathcal{S}_p)$ is the c-degree β_{k+2-i} that corresponds to the p-degree α_i of the smallest possible world t_i of t in which σ is violated, that is,*

$$c_t(\sigma) = \begin{cases} \beta_1 & , \textit{ if } \models_{t_k} \sigma \\ \min\{\beta_{k+2-i} | \not\models_{t_i} \sigma\} & , \textit{ else}; \end{cases}$$

For example, when t denotes the p-SQL table of Table 1, then $c_t(dpt \rightarrow mng) = \beta_1$, $c_t(emp \rightarrow dpt) = \beta_2$, $c_t(u(emp)) = \beta_3$, and $c_t(n(emp)) = \beta_4$.

The primary use of constraints is to specify the semantics of an application domain, that is, to stipulate which databases are regarded as meaningful in the context of the application and which are not. According to this view, we classify a possibilistic SQL table as meaningful whenever it satisfies a given set of possibilistic constraints. For that purpose, a possibilistic constraint (σ, β) (key, FD, `NOT NULL` constraint) allows us to stipulate the minimum marginal c-degree β by which the constraint σ must hold in every p-SQL table that is considered to be meaningful in the application domain.

Definition 2 (Possibilistic constraints) *Let $(T, \mathcal{S}_P)$ denote a p-SQL table schema. A* possibilistic SQL key, possibilistic SQL FD, *or* possibilistic `NOT NULL` constraint *is a pair*

(σ,β) where σ denotes an SQL key, FD or `NOT NULL` *constraint over the table schema T, respectively, and β denotes a c-degree from $\mathcal{S}_P^T$. The p-constraint (σ,β_i) is said to be* satisfied *by a p-SQL table t over $(T,\mathcal{S}_P)$ if and only if the marginal c-degree of σ in t is β_i or higher, that is, $c_t(\sigma)\geq\beta_i$.*

For example, when t denotes the p-SQL table of Table 1, then the following examples of p-constraints are satisfied by t: $(dpt\to mng,\beta_3)$ since $c_t(dpt\to mng)=\beta_1\geq\beta_3$, $(emp\to dpt,\beta_2)$ since $c_t(emp\to dpt)=\beta_2\geq\beta_2$, and $(u(emp),\beta_4)$ since $c_t(u(emp))=\beta_3\geq\beta_4$. In other words, t satisfies these three constraints. On the other hand, t violates (i.e. does not satisfy) any of the following p-constraints: $(emp\to dpt,\beta_1)$ since $c_t(emp\to dpt)=\beta_2<\beta_1$, $(u(emp),\beta_2)$ since $c_t(u(emp))=\beta_3<\beta_2$, and $(n(emp),\beta_3)$ since $c_t(n(emp))=\beta_4<\beta_3$.

5 Motivating the Core Reasoning Problems

We illustrate the centrality of the implication problem based on the two most common database tasks: updates and queries.

5.1 Efficient updates

Consider our p-table schema $(\text{Work},\alpha_1>\alpha_2>\alpha_3>\alpha_4)$ and suppose a design team has specified the set Σ_1 of the following p-constraints that shall be enforced by our database system: $(emp\to dpt,\beta_2)$, $(dpt\to mng,\beta_1)$, $(u(emp),\beta_3)$, $(n(dpt,mng),\beta_1)$, and $(u(mng),\beta_2)$. Since p-tables are typically large, it is important not to delay any insertions into the table without good reason. Since the database system must validate that the updated table does not violate any of the p-constraints in Σ_1, we want to ensure that none of these checks are redundant. For example, validating that $(u(emp),\beta_3)$ is redundant if we already know that the updated table satisfies all the other constraints in Σ_1. Here, redundancy means that $\sigma=(u(emp),\beta_3)$ is implied by $\Sigma_1-\{\sigma\}$: every p-table that satisfies $\Sigma_1-\{\sigma\}$ also satisfies σ. Hence, if we can decide at design time whether a given set of p-constraints implies another p-constraint, then we can avoid any redundant validation checks at run time of the database. This saves more time the more rows our table contains.

5.2 Query optimization

Consider the same p-table schema $(\text{Work},\alpha_1>\alpha_2>\alpha_3>\alpha_4)$ and our set Σ of p-constraints: $(emp\to dpt,\beta_2)$, $(dpt\to mng,\beta_1)$, $(u(emp),\beta_3)$. The SQL query Q_1 on the left of Table 4 selects all distinct employees from rows with p-degree α_2. When evaluated on the p-table t in Table 1, then the answers 'John' and 'Andy' are returned. Removing the keyword `DISTINCT` from the query Q_1 would lead to the following collection of answers: 'John', 'Andy', and 'Andy', in which duplicate answers are not removed. In general, duplicate removal must be explicitly specified by users of SQL as the default is not to remove duplicates. The reason is that duplicates are important for aggregation

Table 4: SQL Queries Q_1, Q_2, and Q_2'

`SELECT DISTINCT` *emp*	`SELECT DISTINCT` *emp*	`SELECT` *emp*
`FROM` WORK	`FROM` WORK	`FROM` WORK
`WHERE` *p-degree*=‘α_2’	`WHERE` *p-degree*=‘α_1’	`WHERE` *p-degree*=‘α_1’

queries, and duplicate removal is an expensive operation. Now consider the SQL query Q_2 in the middle of Table 4 which selects all distinct employees from rows with p-degree α_1. Here, the clause `DISTINCT` is redundant. That is, the query Q_2' on the right of Table 4 returns the same answers as Q_2 over every legal table. Here, legal means that the table satisfies the given constraints in Σ. However, Q_2' is potentially more efficient to run than Q_2 as the latter will (without any chance of success) attempt to look for duplicates and remove them. The reason is the p-key $(u(emp), \beta_3)$, which says that there are no two different rows with p-degree α_1 that have matching non-null values on *emp*. Hence, a query optimizer that can infer the maximum c-degree by which a given p-constraint is implied by a given set of p-constraints is able to optimize SQL queries. In our example, the maximum c-degree by which $u(emp)$ is implied by Σ is β_3, meaning that Q_2 can be rewritten into the equivalent query Q_2' but removal of `DISTINCT` from Q_1 will result in a non-equivalent query. Again, this highlights the centrality of the implication problem.

5.3 The implication problem

Broadly stated, the (finite) implication problem for a class $\mathcal{C}$ of constraints is to decide for any given finite set $\Sigma \cup \{\varphi\}$ in $\mathcal{C}$ whether Σ (finitely) implies φ. That is, whether every (finite) database instance that satisfies all constraints in Σ also satisfies φ. For our purposes, the given set Σ of constraints will always be defined over a single p-SQL table schema, and as the p-scale is finite, so is Σ. Furthermore, in our context we are interested in the class $\mathcal{C}$ of p-SQL keys, FDs, and `NOT NULL` constraints. For this class, the finite implication problem and the implication problem coincide. In particular, if there is an infinite p-SQL table that satisfies all elements of Σ but violates φ, then - by the semantics of our constraints - there must be two rows in that table that violate φ. However, those two rows satisfy all elements of Σ, so there is also a finite p-SQL table showing that Σ does not finitely imply φ. In the remainder we will therefore speak of *the* implication problem. We write $\Sigma \models \varphi$, if Σ implies φ. The problem is defined as follows:

PROBLEM:	IMPLICATION
INPUT:	P-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$
	Set $\Sigma \cup \{\varphi\}$ of p-SQL keys, FDs and `NOT NULL` constraints
OUTPUT:	Yes, if $\Sigma \models \varphi$
	No, otherwise

Consider the p-SQL table schema $(\text{WORK}, \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4)$ with the following set Σ of p-constraints: $(dpt \to mng, \beta_1)$, $(emp \to dpt, \beta_2)$, and $(u(emp), \beta_3)$. Then Σ does not imply $(emp \to mng, \beta_2)$, as Table 1 shows. If Θ results from Σ by adding the `NOT NULL` constraint $(n(dpt), \beta_3)$, then Θ still does not imply $(emp \to mng, \beta_2)$, as rows 8

Table 5: Axiomatization $\mathfrak{pS}$ of p-SQL keys, FDs, and `NOT NULL` constraints

$\dfrac{}{(XY \to X, \beta_i)}$ (FD reflexivity)	$\dfrac{(X \to YZ, \beta_i)}{(X \to Y, \beta_i)}$ (FD decomposition)	$\dfrac{(X \to Y, \beta_i)\ (Y \to Z, \beta_i)}{(X \to Z, \beta_i)} Y \subseteq XT_s^i$ (null FD transitivity)
$\dfrac{(X \to Y, \beta_i)\ (X \to Z, \beta_i)}{(X \to YZ, \beta_i)}$ (FD union)	$\dfrac{(uX, \beta_i)}{(X \to Y, \beta_i)}$ (key weakening)	$\dfrac{(X \to Y, \beta_i)\ (uY, \beta_i)}{(uX, \beta_i)} Y \subseteq XT_s^i$ (null pullback)
$\dfrac{(\sigma, \beta_i)}{(\sigma, \beta_{i+1})}$ (submodel)	$\dfrac{}{(\sigma, \beta_{k+1})}$ (triviality)	

and 9 of Table 1 show. However, if Ω results from Σ by adding the `NOT NULL` constraint $(n(dpt), \beta_2)$, then Ω does imply $(emp \to mng, \beta_2)$.

6 Axiomatic Characterization

We will now present an axiomatic, algorithmic, and logical characterization for the implication problem of our p-constraint class, illustrate them by examples, and highlight important consequences. For their detailed proofs we need to establish additional results about the non-possibilistic class. The details of the proofs and the additional results can be found in a technical report.

The set $\Sigma^* = \{\varphi \mid \Sigma \models \varphi\}$ denotes the *semantic closure of* Σ, that is, the set of all (p-)constraints that are implied by Σ. In principle, the definition of the semantic closure does not tell us whether we can compute it, nevermind how. It is a core reasoning task to investigate whether/how a semantic notion can be characterized syntactically. In fact, we determine the semantic closure Σ^* of a set Σ of (p-)constraints by applying *inference rules* or *axioms* of the form $\dfrac{\text{premise}}{\text{conclusion}}\text{condition}$. For a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\sigma_1, \ldots, \sigma_n$ such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or is the conclusion that results from an application of an inference rule in $\mathfrak{R}$ to some premises in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. Let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ be the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. $\mathfrak{R}$ is *sound* (*complete*) if for every (p-)SQL table schema, and for every set Σ of (p-)SQL constraints over this schema we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is a (finite) *axiomatization* if $\mathfrak{R}$ is both sound and complete.

Our first main result is that the set $\mathfrak{pS}$ of inference rules from Table 5 forms a finite axiomatization for the implication problem of our combined class of p-SQL keys, FDs, and `NOT NULL` constraints. For the set of inference rules we assume that X, Y, Z denote attribute subsets of T from the given p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$, and for $i = 1, \ldots, k$, T_s^i denotes the attribute subset of T whose elements belong to some (nX, β_j) where $j \leq i$, that is, the attributes declared `NOT NULL` with c-degree β_i or

Table 7: Axiomatization $\mathfrak{S}$ of SQL keys, FDs, and `NOT NULL` constraints

$\dfrac{}{XY \to X}$ (FD reflexivity)	$\dfrac{X \to YZ}{X \to Y}$ (FD decomposition)	$\dfrac{X \to Y \quad Y \to Z}{X \to Z}\, Y \subseteq XT_s$ (null FD transitivity)
$\dfrac{X \to Y \quad X \to Z}{X \to YZ}$ (FD union)	$\dfrac{uX}{X \to Y}$ (key weakening)	$\dfrac{X \to Y \quad uY}{uX}\, Y \subseteq XT_s$ (null pullback)

higher: $T_s^i = \{A \in T \mid \exists(nX, \beta_j) \in \Sigma \wedge A \in X \wedge j \leq i\}$. Then $\Sigma \models (nX, \beta_i)$ if and only if $X \subseteq T_s^i$. Finally, σ may denote any SQL key or FD over T.

Theorem 1 *The set* $\mathfrak{pS}$ *forms a finite axiomatization for the implication problem of the combined class of p-SQL keys, FDs, and* `NOT NULL` *constraints.*

The main ideas for the proof of Theorem 1 are as follows. Firstly, we establish the first axiomatization $\mathfrak{S}$ for SQL keys, FDs, and `NOT NULL` constraints in the non-possibilistic case. The rules in $\mathfrak{S}$ results from those in $\mathfrak{pS}$ by removing *submodel* and *triviality*, removing the p-degrees β_i from all remaining rules in $\mathfrak{S}$, and replacing T_s^i by T_s in the conditions of *null FD transitivity* and *null pullback*. The rules of $\mathfrak{S}$ are shown in Table 7.

Table 6: Axiomatization $\mathfrak{B}$ of keys and FDs over bag schemata

$\dfrac{}{XY \to X}$ (FD reflexivity)	
$\dfrac{X \to Y}{X \to XY}$ (FD extension)	$\dfrac{X \to Y \quad Y \to Z}{X \to Z}$ (FD transitivity)
$\dfrac{uX}{X \to Y}$ (key weakening)	$\dfrac{X \to Y \quad uY}{uX}$ (pullback)

The completeness proof for $\mathfrak{S}$ uses the completeness of the existing axiomatization $\mathfrak{B}$ from Table 6 for the class of keys and FDs over bag schemata (table schemata where all columns are `NOT NULL`) [14]. Indeed, we can show that every SQL key $\varphi = uX$ and every SQL FD $\varphi = X \to Y$ that is implied by a set Σ of SQL keys and FDs over the table schema T with NFS T_s, φ is also implied by the *XT_s-guarded FD set*

$$\Sigma[XT_s] \;=\; \{V \to W \in \Sigma \mid V \subseteq XT_s\} \cup \{uV \in \Sigma \mid V \subseteq XT_s\}$$

over the bag schema T. The completeness of $\mathfrak{B}$ entails that $\Sigma[XT_s] \vdash_{\mathfrak{B}} \varphi$ holds, and we can show that this implies $\Sigma \vdash_{\mathfrak{S}} \varphi$, establishing completeness of $\mathfrak{S}$. Secondly, we can establish the completeness of $\mathfrak{pS}$ by the completeness of $\mathfrak{S}$ using the *β_i-cut*

$$\Sigma_i = \{\sigma \mid \exists(\sigma, \beta) \in \Sigma \wedge \beta \geq \beta_i\}$$

for a p-constraint set Σ. Indeed, given a set $\Sigma \cup \{(\varphi, \beta_i)\}$ of p-SQL keys, functional dependencies, and `NOT NULL` constraints over a p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$ where $1 \leq i \leq k$, we can show that $\Sigma \models (\varphi, \beta_i)$ if and only if $\Sigma_i \models \varphi$. Consequently, given a p-SQL key or p-SQL FD $\varphi = (\sigma, \beta_i)$ that is implied by a set Σ of p-SQL keys, p-SQL FDs, and p-SQL `NOT NULL` constraints over p-SQL schema T, it follows that the SQL key $\sigma = uX$ or SQL FD $\sigma = X \to Y$ is implied by the SQL key/FD set Σ_i over SQL table schema T with NFS T_s^i. By the completeness of $\mathfrak{S}$ we obtain $\Sigma_i \vdash_{\mathfrak{S}} \sigma$, and we can then show that also $\Sigma \vdash_{\mathfrak{pS}} (\sigma, \beta_i) = \varphi$. This shows the completeness of $\mathfrak{pS}$.

Consider the p-table schema $(\text{Work}, \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4)$ with the following set Σ of p-constraints: $(dpt \to mng, \beta_1)$, $(emp \to dpt, \beta_2)$, and $(u(emp), \beta_3)$. Table 1, in particular rows 8 and 9, shows that Σ does not imply $\varphi = (emp \to mng, \beta_2)$. However, if Ω results from Σ by adding the `NOT NULL` constraint $(n(dpt), \beta_2)$, then Ω does imply $(emp \to mng, \beta_2)$. Using Theorem 1 this is equivalent to showing that $(emp \to mng, \beta_2)$ can be inferred from Ω using $\mathfrak{pS}$. Indeed, we can apply the *submodel* rule to $(dpt \to mng, \beta_1)$ to infer $(dpt \to mng, \beta_2)$. Subsequently, we can apply the *null FD transitivity* rule to $(emp \to dpt, \beta_2)$, $(dpt \to mng, \beta_2)$, and $T_s^2 = \{dpt\}$ to infer $(emp \to mng, \beta_2)$.

7 Algorithmic Characterization

We use the main proof arguments outlined after Theorem 1 to reduce the possibilistic implication problem to the standard computation of an attribute set closure in the relational model of data. This constitutes a clean solution that makes effective use of already existing research, and, as an outcome, results in an efficient computation in linear time.

As usual, for an attribute subset $X \subseteq T$ and an FD set Σ, let $X_\Sigma^+ = \{A \in T \mid \Sigma \models X \to A\}$ be the attribute set closure of X under Σ [7]. The final reduction is achieved by the *FD reduct* $\Sigma[\text{FD}] = \{V \to W \mid V \to W \in \Sigma\} \cup \{V \to T \mid uV \in \Sigma\}$ for a set Σ of keys and FDs over bag schema T. Given these definitions we can now state the following result.

Theorem 2 *Let Σ denote a set of p-SQL keys, FDs, and* `NOT NULL` *constraints over p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$. For all $(X \to Y, \beta_i)$, (uX, β_i) and (nX, β_i) over $(T, \alpha_1 > \cdots > \alpha_{k+1})$ the following hold:*

1. *$\Sigma \models (X \to Y, \beta_i)$ if and only if $Y \subseteq X^+_{\Sigma_i[XT_s^i][FD]}$.*
2. *$\Sigma \models (uX, \beta_i)$ if and only if $X^+_{\Sigma_i[XT_s^i][FD]} = T$ and there is some $uZ \in \Sigma_i[XT_s^i]$.*
3. *$\Sigma \models (nX, \beta_i)$ if and only if $X \subseteq T_s^i$.*

Consider the p-SQL table schema $(\text{Work}, \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4)$ with the previous set Ω of p-constraints. While Ω does imply $(emp \to mng, \beta_2)$, Ω does not imply $(u(emp), \beta_2)$. Indeed, $\Omega_2[emp,dpt] = \{emp \to dpt, dpt \to mng\}$. Hence, $emp^+_{\Omega_2[emp,dpt]} = \text{Work}$, but there is no $uY \in \Omega_2[emp,dpt]$. Indeed, rows 6 and 7 from Table 1 form a two-row table that satisfies Ω but not $(u(emp), \beta_2)$.

According to Theorem 2 deciding instances of the possibilistic implication problem are effectively reduced to the standard computation of the attribute set closure $X^+_{\Sigma_i[XT_s^i][FD]}$ under the set $\Sigma_i[XT_s^i][FD]$ of FDs. Algorithms that work in linear input time for this computation are well-known [7]. As a summary of our reductions, we present here Algorithm 1 that computes $X^+_{\Sigma_i[XT_s^i][FD]}$ as the attribute set closure $\text{Closure}(T, \Sigma, i, X)$ of X with respect to the set Σ of p-SQL keys, FDs and `NOT NULL` constraints.

Algorithm 1 $\textsc{Closure}(T, \Sigma, i, X)$ - Attribute Set Closure

```
Require: p-SQL table schema $T$, set $\Sigma$ of p-SQL keys, FDs, and NOT NULL constraints
    over $T$, c-degree $\beta_i$, set $X \subseteq T$
Ensure: Attribute set closure $X^+_{\Sigma_i[XT^i_s][FD]}$ of $X$ wrt set $\Sigma_i[XT^i_s][FD]$ of FDs over $T$
 1: $Closure \leftarrow X$;
 2: $FDList \leftarrow$ List of FDs in $\Sigma_i[XT^i_s][FD]$;
 3: repeat
 4:     $OldClosure \leftarrow Closure$;
 5:     Remove all attributes in Closure from LHS of FDs in FDlist;
 6:     for all $\{\emptyset \rightarrow Y\} \in FDList$ do
 7:         $Closure \leftarrow Closure \cup Y$;
 8:         $FDList \leftarrow FDList - \{\emptyset \rightarrow Y\}$;
 9: until $Closure = OldClosure$ or $FDList = \emptyset$ or $Closure = T$
10: return $Closure$
```

Algorithm 1 shows that each attribute that occurs in the input is visited at most once. We therefore obtain the following linear-time complexity, which follows from results in [7]. Here, we define the size $||\Sigma||$ of a set Σ of p-constraints as the total number of attribute occurrences in Σ.

Theorem 3 *The instance (Σ, φ) of the implication problem for the class of p-SQL keys, FDs, and* `NOT NULL` *constraints over p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$ can be decided in time in $\mathcal{O}(\max\{|T|, ||\Sigma \cup \{\varphi\}||\})$.* ∎

8 Logical Characterization via HornSAT

In this section we put our and previous results together to realize that we can reason about p-keys and p-FDs over possibilistic SQL tables very efficiently and with very familiar tools. Hence, our formalizations of keys and FDs as well as those of the variety and veracity dimensions of big data have led us to a framework where we can express integrity constraint for entities in big data sets, and can reason about these entities confidently and efficiently.

Following the outline of the main proof arguments after Theorem 1, we have reduced instances $\Sigma \models (\sigma, \beta_i)$ for keys $\sigma = uX$ or FDs $\sigma = X \rightarrow Y$ of the implication problem for our class of p-constraints to instances $\Sigma_i[XT^i_s] \models \sigma$ for the implication problem of keys and FDs over bag schemata. For the latter it is known that keys behave like propositional goal clauses and FDs behave like propositional definite clauses [8, 14].

More formally, we map a key $\sigma = u(A_1, \ldots, A_n)$ over T to the goal clause $\sigma' = \neg A'_1 \vee \cdots \neg A'_n$ over the set $T' = \{A' \mid A \in T\}$ of propositional variables that correspond to attributes in T, and we map an FD $\sigma = A_1, \ldots, A_n \rightarrow A$ to a definite clause $\sigma' = \neg A'_1 \vee \cdots \neg A'_n \vee A'$ over the set T'. If $\models_2$ denotes classical propositional entailment, then it is known that $\Sigma \models \varphi$ if and only if $\Sigma' = \{\sigma' \mid \sigma \in \Sigma\} \models_2 \varphi'$.

Theorem 4 *The implication problem of p-SQL keys, FDs, and* `NOT NULL` *constraints is* PTIME*-complete and equivalent to* $\textsc{Horn}$SAT*.* ∎

Concluding our illustrations on our running example, reconsider the instances $\Omega \models (emp \to mng, \beta_2)$ and $\Omega \models (u(emp), \beta_2)$ of the implication problem over p-SQL table schema WORK. Recall that $\Omega_2[emp,dpt] = \{emp \to dpt, dpt \to mng\}$. Consequently, $(\Omega_2[emp,dpt])' = \{\neg emp' \vee dpt', \neg dpt' \vee mng'\}$. Applying linear resolution, for example, it is easy to see that the Horn formulae set $(\Omega_2[emp,dpt])' \cup \{emp', \neg mng'\}$ is unsatisfiable, while the Horn formulae set $(\Omega_2[emp,dpt])' \cup \{emp'\}$ is satisfiable by interpreting all atoms as `true`. Indeed, the latter propositional model corresponds to the two-row table with row 6 and 7 from Table 1, which shows that Ω does not imply $(u(emp), \beta_2)$. This is also an illustration of the fact that our correspondence does not just hold between the implication problems of constraints and Horn clauses, but extends to counter-example two-row tables and truth assignments.

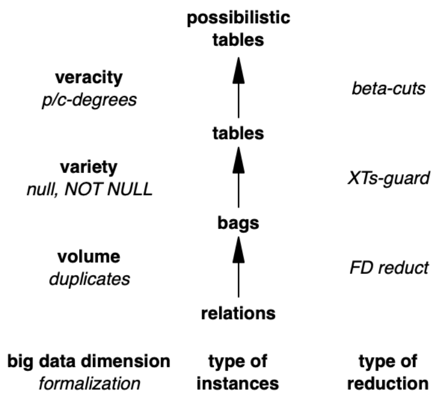


Figure 2: Summary

9 Conclusion

We have established a comprehensive toolbox for automated reasoning about the data integrity of real-world entities in big data sets, including axiomatic, algorithmic and logical characterizations. These rely on specific formalizations of integrity constraints, the variety and the veracity dimension of big data. As an underlying data model we chose a possibilistic extension of the de-facto industry standard SQL for data management. With adequate definitions it turns out that automated reasoning can be done by applying off-the-shelf tools, such as linear resolution for propositional HORNSAT.

Our work suggests a whole landscape of future work. Here, different approaches should be applied to the big data dimensions, such as probabilistic approaches to the veracity dimension, different approaches of handling missing information to the variety dimension, and different approaches to the formalization of entity integrity such as key sets [11] or embedded keys and FDs [24].

References

[1] P. Atzeni and N. M. Morfuni. Functional dependencies and constraints on null values in database relations. *Information and Control*, 70(1):1–31, 1986.

[2] C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Trans. Database Syst.*, 4(1):30–59, 1979.

[3] L. E. Bertossi. *Database Repairing and Consistent Query Answering.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[4] V. Christophides, V. Efthymiou, and K. Stefanidis. *Entity Resolution in the Web of Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2015.

[5] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.

[6] C. J. Date. A critique of the SQL database language. *SIGMOD Record*, 14(3):8–54, 1984.

[7] J. Diederich and J. Milton. New methods and fast algorithms for database normalization. *ACM Trans. Database Syst.*, 13(3):339–365, 1988.

[8] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.

[9] D. Dubois, H. Prade, and S. Schockaert. Generalized possibilistic logic: Foundations and applications to qualitative reasoning about uncertainty. *Artif. Intell.*, 252:139–174, 2017.

[10] V. Ganti and A. D. Sarma. *Data Cleaning: A Practical Perspective.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[11] M. Hannula and S. Link. Automated reasoning about key sets. In *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, pages 47–63, 2018.

[12] S. Hartmann and S. Link. When data dependencies over SQL tables meet the logics of paradox and S-3. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 317–326, 2010.

[13] S. Hartmann and S. Link. The implication problem of data dependencies over SQL table definitions: Axiomatic, algorithmic and logical characterizations. *ACM Trans. Database Syst.*, 37(2):13:1–13:40, 2012.

[14] H. Köhler and S. Link. Armstrong axioms and boyce-codd-heath normal form under bag semantics. *Inf. Process. Lett.*, 110(16):717–724, 2010.

[15] Y. E. Lien. On the equivalence of database models. *J. ACM*, 29(2):333–362, 1982.

[16] S. Link and H. Prade. Possibilistic functional dependencies and their relationship to possibility theory. *IEEE Trans. Fuzzy Systems*, 24(3):757–763, 2016.

[17] S. Link and H. Prade. Relational database schema design for uncertain data. *Inf. Syst.*, 84:88–110, 2019.

[18] Z. H. Liu, B. C. Hammerschmidt, and D. McMahon. JSON data management: supporting schema-less development in RDBMS. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1247–1258, 2014.

[19] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[20] A. Pavlo and M. Aslett. What's really new with NewSQL? *SIGMOD Record*, 45(2):45–55, 2016.

[21] O. Pivert and H. Prade. Handling uncertainty in relational databases with possibility theory - A survey of different modelings. In *Scalable Uncertainty Management - 12th International Conference, SUM 2018, Milan, Italy, October 3-5, 2018, Proceedings*, pages 396–404, 2018.

[22] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[23] B. Thalheim. *Dependencies in relational databases.* Teubner, 1991.

[24] Z. Wei and S. Link. Embedded functional dependencies and data-completeness tailored database design. *PVLDB*, 12(11):1458–1470, 2019.

[25] S. Yin and O. Kaynak. Big data for modern industry: Challenges and trends. *Proceedings of the IEEE*, 103(2):143–146, 2015.

[26] C. Zaniolo. Database relations with null values. *J. Comput. Syst. Sci.*, 28(1):142–166, 1984.

A SQL Constraints

In this section of the appendix we will develop axiomatic and algorithmic characterizations for the implication problem of the combined class of SQL keys, FDs and `NOT NULL` constraints.

A.1 Axiomatic Characterization

Our first goal is to show that the set $\mathfrak{S}$ of inference rules in Table 7 form an axiomatization for the combined class of keys, FDs and `NOT NULL` constraints. FD reflexivity axiom, FD decomposition, null FD transitivity, and FD union rule are the familiar rules from Atzeni and Morfuni's axiomatization of FDs under the 'no information' interpretation of null marker occurrences [1,13]. The key weakening rule says that every SQL table that satisfies a key uX also satisfies any FD with left-handed side X. Furthermore, the null pullback rule captures additional non-trivial interactions of keys and FDs over SQL tables with an NFS. We show first that these rules are sound.

Lemma 1 *The inference rules in $\mathfrak{S}$ are sound for the implication of keys and FDs over SQL tables with NFSs.*

Proof We exhibit the soundness for each of the six rules.

For *FD reflexivity* let t be an arbitrary SQL table over T, and $XY \to X$ an FD over T. Trivially, any two rows $r_1, r_2 \in t$ that have matching non-null values on all the columns in X and Y also have matching values on all the columns in X.

For *FD decomposition* let t be an arbitrary SQL table over T such that t violates the FD $X \to Y$ over T. Hence, there are two rows $r_1, r_2 \in t$ that have matching non-values on all the attributes in X but non-matching values on some attribute in Y. Consequently, $t_1, t_2 \in r$ have matching non-values on all the attributes in X but non-matching values on some attribute in YZ. Consequently, t also violates the FD $X \to YZ$ over T.

For *null FD transitivity* let t be an arbitrary SQL table over T such that t violates the FD $X \to Z$ over T, satisfies the FD $X \to Y$ over T and satisfies the condition $Y \subseteq XT_s$ where $nfs(T_s)$ is an NFS over T. Hence, there are two rows $r_1, r_2 \in t$ that have matching non-null values on all the attributes in X but non-matching values on some attribute in Z. Since t satisfies the FD $X \to Y$, $r_1, r_2 \in t$ must have matching values on all the attributes in Y. However, since t satisfies $Y \subseteq XT_s$, it even follows that $r_1, r_2 \in t$ must have matching non-null values on all the attributes in Y. Since $r_1, r_2 \in t$ have non-matching values on some attribute in Z, we conclude that t violates the FD $Y \to Z$.

For the *union rule* let t be an arbitrary SQL table over T such that t violates the FD $X \to YZ$ over T. Hence, there are two rows $r_1, r_2 \in t$ with matching non-null values on all the attributes in X but non-matching values on some attribute in YZ. Consequently, $r_1, r_2 \in t$ have non-matching values on some attribute Y or non-matching values on some attribute in Z. Thus, t violates the FD $X \to Y$ or the FD $X \to Z$.

For *key weakening* let t be an arbitrary SQL table over T such that t violates the FD $X \to Y$ over T. Hence, there are two rows $r_1, r_2 \in t$ with matching non-null values on all the attributes in X and non-matching values on some attribute in Y. The latter condition implies that r_1 and r_2 must be different rows. Hence, the key uX is violated by t.

For *null pullback* let t be an arbitrary SQL table over T such that t violates the key uX over T, satisfies the FD $X \to Y$ over T and satisfies the condition $Y \subseteq XT_s$ where $\mathit{nfs}(T_s)$ is an NFS over T. Hence, there are two different rows $r_1, r_2 \in t$ that have matching non-null values on all the attributes in X. Since t satisfies the FD $X \to Y$, the rows $r_1, r_2 \in t$ have matching values on all the attributes in Y. However, since t satisfies $Y \subseteq XT_s$, it even follows that the rows $r_1, r_2 \in t$ have matching non-null values on all the attributes in Y. We conclude that t violates the key uY.

Let $\mathfrak{F}$ consist of FD reflexivity, FD decomposition, null FD transitivity, and FD union from Table 7. For the reason of being self-contained we provide a completeness proof here. We use a classical proof technique to establish the completeness of $\mathfrak{F}$ for the implication of FDs over SQL tables with an NFS.

Theorem 5 *The set $\mathfrak{F}$ is a finite axiomatization for the implication of FDs over SQL tables with an NFS.*

Proof Let T be an SQL table schema, Σ a set of FDs and $\mathit{nfs}(T_s)$ an NFS over T. Let $X \to Y \notin \Sigma^+_{\mathfrak{F}}$. We establish the completeness of $\mathfrak{F}$ by showing that $X \to Y \notin \Sigma^*_{T_s}$. That is, we construct an SQL table t over T that is T_s-total, satisfies Σ and violates $X \to Y$. Let $X^+_{\mathfrak{F}} = \{A \in T \mid \Sigma \vdash_{\mathfrak{F}} X \to A\}$. Consequently, $Y \not\subseteq X^+_{\mathfrak{F}}$ and there is some $A_0 \in Y - X^+_{\mathfrak{F}}$. Indeed, if $Y \subseteq X^+_{\mathfrak{F}}$, then $X \to Y \in \Sigma^+_{\mathfrak{F}}$ by *FD union*. Let $t := \{r_1, r_2\}$ be the SQL table over T where r_1, r_2 are defined by:

$X^+_{\mathfrak{F}} \cap XT_s$	$(T_s - X^+_{\mathfrak{F}})A_0$	Otherwise
$0 \cdots 0$	$0 \cdots 0$	$\perp \cdots \perp$
$0 \cdots 0$	$1 \cdots 1$	$\perp \cdots \perp$

.

The table t violates the FD $X \to Y$ since $A_0 \in Y$. Furthermore, t is T_s-total. It remains to show that t satisfies Σ. Let $U \to V \in \Sigma$ and let r_1, r_2 have matching non-null values on all the attributes in U. By construction, $U \subseteq X^+_{\mathfrak{F}}$. Hence, $X \to U \in \Sigma^+_{\mathfrak{F}}$ holds by the definition of $X^+_{\mathfrak{F}}$. Furthermore, by construction, $U \subseteq XT_s$. From $X \to U \in \Sigma^+_{\mathfrak{F}}$, $U \to V \in \Sigma$, and $U \subseteq XT_s$, it follows by *null FD transitivity* that $X \to V \in \Sigma^+_{\mathfrak{F}}$. Thus, $V \subseteq X^+_{\mathfrak{F}}$. In particular, $A_0 \notin V$ since otherwise $A_0 \in X^+_{\mathfrak{F}}$ by *FD decomposition.* This means that r_1, r_2 have matching values on all the attributes in V. Consequently, t satisfies $U \to V \in \Sigma$.

Next we establish the completeness of $\mathfrak{G}$ for the implication of the combined class of keys and FDs over SQL tables with NFS by a reduction to the completeness of $\mathfrak{F}$. Two preparatory lemmata are required first.

For a set Σ of keys and FDs over SQL table schema T let

$$\Sigma[\mathrm{FD}] = \{X \to T \mid uX \in \Sigma\} \cup \{X \to Y \mid X \to Y \in \Sigma\}$$

denote the *FD-reduct* of Σ. Our first lemma says that an FD is implied by a key/FD-set if and only if it is implied by the FD-reduct of the set.

Lemma 2 *Let $\Sigma \cup \{X \to Y\}$ be a set of keys and FDs, and nfs(T_s) an NFS over table schema T. Then $\Sigma \models_{T_s} X \to Y$ if and only if $\Sigma[FD] \models_{T_s} X \to Y$.*

Proof If $\Sigma[\text{FD}] \models_{T_s} X \to Y$, then $\Sigma \models_{T_s} X \to Y$ because *key weakening* is sound.

If $\Sigma[\text{FD}] \not\models_{T_s} X \to Y$, then consider the two-row table t from the completeness proof of Theorem 5. In particular, t satisfies $\Sigma[\text{FD}]$, t is T_s-total and t violates $X \to Y$. Moreover, t satisfies every key $uZ \in \Sigma$: otherwise, the two rows $r_1, r_2 \in t$ would be duplicates as $Z \to T \in \Sigma[\text{FD}]$ is satisfied by t, and t would also satisfy $X \to Y$, a contradiction. Consequently, t shows that $\Sigma \not\models_{T_s} X \to Y$.

Our second lemma says that a key uX is implied by a key/FD set if and only if the key attributes functionally determine all attributes by the FD-reduct and some key exists in the key/FD set where all of its attributes cannot contain a null marker occurrences or belong to X.

Lemma 3 *Let $\Sigma \cup \{uX\}$ be a set of keys and FDs, and nfs(T_s) an NFS over SQL table schema T. Then $\Sigma \models_{T_s} uX$ if and only if $\Sigma[FD] \models_{T_s} X \to T$ and there is some $uZ \in \Sigma$ such that $Z \subseteq XT_s$.*

Proof Suppose $\Sigma[\text{FD}] \models_{T_s} X \to T$ and there is some $uZ \in \Sigma$ such that $Z \subseteq XT_s$. The soundness of *FD decomposition* shows that $\Sigma[\text{FD}] \models_{T_s} X \to Z$. The soundness of *null pullback* shows that $\Sigma[\text{FD}] \models_{T_s} uX$. Finally, the soundness of *key weakening* shows that $\Sigma \models_{T_s} uX$.

Suppose that $\Sigma \models_{T_s} uX$. The soundness of *key weakening* means that $\Sigma \models_{T_s} X \to T$ holds, too. From Lemma 2 we conclude that $\Sigma[\text{FD}] \models_{T_s} X \to T$ holds. It remains to show that if $\Sigma \models_{T_s} uX$ holds, then there is some $uZ \in \Sigma$ such that $Z \subseteq XT_s$. We show the contraposition: if $Z \not\subseteq XT_s$ holds for all $uZ \in \Sigma$, then $\Sigma \not\models_{T_s} uX$. Indeed, let t' be the SQL table consisting of the following two rows:

XT_s	$T - (XT_s)$
$0 \cdots 0$	$\bot \cdots \bot$
$0 \cdots 0$	$\bot \cdots \bot$

.

Trivially, t' satisfies every FD and is T_s-total. For any $uZ \in \Sigma$ we know that $Z \not\subseteq XT_s$ holds. Hence, t' also satisfies every key in Σ, and thereby, Σ itself. Since t' violates uX, it follows that $\Sigma \not\models_{T_s} uX$.

We can now establish the completeness of $\mathfrak{S}$.

Theorem 6 *The set $\mathfrak{S}$ is a finite axiomatization for the implication of keys and FDs in the presence of an NFS.*

Proof Let $\Sigma \cup \{\varphi\}$ be a set of keys and FDs, and *nfs*(T_s) an NFS over SQL table schema T. We exploit Lemmata 2 and 3 and Theorem 5 to show directly that $\Sigma \models_{T_s} \varphi$ entails $\Sigma \vdash_{\mathfrak{S}} \varphi$ holds.

Suppose that φ denotes the FD $X \to Y$. From Lemma 2 we conclude that $\Sigma[\text{FD}] \models_{T_s} X \to Y$ holds. As $\mathfrak{F}$ is complete for the implication of FDs over SQL tables with

NFS, it follows that $\Sigma[\text{FD}] \vdash_{\mathfrak{F}} X \to Y$ holds. Since $\mathfrak{F} \subseteq \mathfrak{S}$ we conclude further that $\Sigma[\text{FD}] \vdash_{\mathfrak{S}} X \to Y$ holds. Finally, *key weakening* guarantees that $\Sigma \vdash_{\mathfrak{S}} \sigma$ holds for all $\sigma \in \Sigma[\text{FD}]$. Hence, $\Sigma \vdash_{\mathfrak{S}} X \to Y$.

Suppose now that φ denotes the p-key uX. From Lemma 3 we conclude that $\Sigma[\text{FD}] \models_{T_s} X \to T$ holds, and that there is some $uZ \in \Sigma$ where $Z \subseteq XT_s$ holds. As in the previous case, it follows that $\Sigma \vdash_{\mathfrak{S}} X \to T$ holds, and the *decomposition rule* shows $\Sigma \vdash_{\mathfrak{S}} X \to Z$. An application of *null pullback* infers that $\Sigma \vdash_{\mathfrak{S}} uX$. This concludes the proof.

We write $\Sigma \models_{2,T_s} \varphi$ to say that φ is implied by Σ and $nfs(T_s)$ over two-row tables. That is, $\Sigma \models_{2,T_s} \varphi$ denotes an instance of the implication problem in the world of two-row tables: every two-row table t over T that is T_s-total and satisfies every element of Σ also satisfies φ. It is a consequence of our previous arguments that $\Sigma \models_{2,T_s} \varphi$ if and only if $\Sigma \models_{T_s} \varphi$ holds. This is an interesting model-theoretic property of our class of constraints. It is a core enabler for establishing a logical characterization of the associated implication problem.

Corollary 1 *For all SQL table schemata T, for all sets $\Sigma \cup \{\varphi\}$ of keys and FDs over T and for all NFSs $nfs(T_s)$ over T, $\Sigma \models_{T_s} \varphi$ if and only if $\Sigma \models_{2,T_s} \varphi$.*

Proof Obviously, if there is some two-row SQL table t over R that is T_s-total, satisfies every element of Σ, and violates φ, then $\Sigma \not\models_{T_s} \varphi$. Vice versa, if $\Sigma \not\models_{T_s} \varphi$, then the proofs of Theorem 5 and Lemma 3 show how to construct a two-row SQL table t over T that is T_s-total, satisfies every element of Σ, and violates φ.

Example 1 *Consider the relation schema* Work *with attributes emp, dpt and mng, NFS* $\text{Work}_s = \{dpt, mng\}$*, and*

$$\Sigma = \{emp \to dpt, dpt \to mng, u(mng)\}.$$

Then $\Sigma \models_{\text{Work}_s} u(emp)$. Indeed, one may apply null FD transitivity to the two FDs of Σ to infer the FD $emp \to mng$. One may then apply null pullback to this FD and the key from Σ to infer the key $u(emp)$. ▮

Example 2 *Consider the SQL table schema* Work *with attributes emp, dpt and mng, NFS* $\text{Work}_s = \{dpt, mng\}$*, and*

$$\Sigma = \{emp \to dpt, dpt \to mng\}.$$

Then $\Sigma \not\models_{\text{Work}_s} u(emp)$, as the SQL table

emp	*dpt*	*mng*
Dilbert	*IT*	*Gates*
Dilbert	*IT*	*Gates*

over Work *with NFS $nfs(\text{Work}_s)$ shows. Note that this SQL table follows the construction in the proof of Lemma 3.* ▮

Example 3 *Consider the SQL table schema* $\textsc{Work}$ *with attributes emp, dpt and mng, NFS* $\textsc{Work}_s = \{dpt\}$*, and*

$$\Sigma = \{emp \to dpt, dpt \to mng, u(mng)\}.$$

Then $\Sigma \not\models_{\textsc{Work}_s} u(emp)$*, as the SQL table*

emp	*dpt*	*mng*
Dilbert	*IT*	$\perp$
Dilbert	*IT*	$\perp$

over $\textsc{Work}$ *with NFS* $nfs(\textsc{Work}_s)$ *shows. Note that this SQL table follows the construction in the proof of Theorem 5.* ∎

A.2 Algorithmic Characterization

Given a table schema T, an NFS $nfs(T_s)$ and a set Σ of keys and FDs over T, the inference system $\mathfrak{S}$ determines the semantic closure $\Sigma^*_{T_s}$. This can take time exponential in the size of the input. In practice, however, knowledge about all elements of the semantic closure is not always required. Instead, it often suffices to know whether some fixed key or FD φ is an element of $\Sigma^*_{T_s}$. In this case one must decide the instance $\Sigma \models_{T_s} \varphi$ of the associated implication problem. For that purpose the computation of $\Sigma^*_{T_s}$ is too expensive. In this section, we develop an algorithm that decides the implication problem efficiently.

Recall the completeness proof of Theorem 5 where we defined the *attribute set closure*

$$X^+_{\mathfrak{F}} := \{A \in T \mid \Sigma \vdash_{\mathfrak{F}} X \to A\}$$

for an attribute set X, and a set Σ of FDs over table schema T with NFS $nfs(T_s)$. Note that Theorem 5 guarantees that $X^+_{\mathfrak{F}} = X^*_{\Sigma,T_s}$ where

$$X^*_{\Sigma,T_s} := \{A \in T \mid \Sigma \models_{T_s} X \to A\}\ .$$

Lemmata 2 and 3 allow us to reduce the implication problem for the combined class of keys and FDs over table schemata with NFSs to the computation of attribute set closures with respect to FDs and an NFS only.

Corollary 2 *Let* Σ *be a set of keys and FDs over table schema* T *with NFS* $nfs(T_s)$*. Then the following holds:*

1. $\Sigma \models_{T_s} X \to Y$ *if and only if* $Y \subseteq X^*_{\Sigma[FD],T_s}$*, and*
2. $\Sigma \models_{T_s} uX$ *if and only if* $X^*_{\Sigma[FD],T_s} = T$ *and there is some* $uZ \in \Sigma$ *such that* $Z \subseteq XT_s$*.*

Corollary 2 motivates Algorithm 2 that computes the attribute set closure $X^*_{\Sigma[FD],T_s}$ for a given attribute set X with respect to a given set Σ[FD] of FDs over a given table schema T with given NFS $nfs(T_s)$.

We show first that Algorithm 2 works correctly, that is, computes indeed the closure of the given attribute set with respect to the given set of FDs and NFS. Note that the proof relies on the completeness of the axiomatization $\mathfrak{F}$.

Algorithm 2 Closure(X, T_s, Σ[FD], T)

```
Require: Attribute set $X$, $nfs(T_s)$, set $\Sigma$[FD] of FDs over $T$
Ensure: Attribute set closure $X^*_{\Sigma,T_s}$
 1: $Closure \leftarrow X$
 2: $FDList \leftarrow$ List of FDs in $\Sigma$[FD]
 3: repeat
 4:     $OldClosure \leftarrow Closure$
 5:     for all $A \in Closure \cap XT_s$ do
 6:         for all $Z \to Y$ in $FDList$ do
 7:             Replace $Z \to Y$ in $FDList$ by $Z - \{A\} \to Y$;
 8:     for all $\emptyset \to Y$ in $FDList$ do
 9:         $Closure \leftarrow Closure \cup Y$
10:         $FDList \leftarrow FDList - \{\emptyset \to Y\}$
11: until $Closure = T$ or $Closure = OldClosure$ or $FDList = [\,]$
12: return($Closure$)
```

Theorem 7 *On input (X, T_s, Σ, T), Algorithm 2 computes the attribute set closure X^*_{Σ,T_s} of X with respect to the FD set Σ over table schema T with NFS nfs(T_s).*

Proof Let *Closure* denote the output of Algorithm 2. It needs to be shown that $Closure = X^*_{\Sigma,T_s}$. Since $X^*_{\Sigma,T_s} = X^+_{\mathfrak{F}}$, this is the same as showing that $Closure = X^+_{\mathfrak{F}}$ holds. For that purpose we proceed in two stages, first showing that $Closure \subseteq X^+_{\mathfrak{F}}$.

We proceed by induction on the number j of runs through the **repeat** loop between lines 3-11 of Algorithm 2. If $j = 0$, then line 1 set *Closure* to X and we have $X \subseteq X^+_{\mathfrak{F}}$ due to the reflexivity axiom. Let $j > 0$. The hypothesis tells us that after j runs we have $Closure \subseteq X^+_{\mathfrak{F}}$. Consider now the $j + 1$st run which adds all attributes of Y to *Closure* whenever there is some FD $Z \to Y \in \Sigma$ where $Z \subseteq Closure$ and $Z \subseteq XT_s$. From $X \to Closure) \in \Sigma^+_{\mathfrak{F}}$ and $Z \subseteq Closure$ we obtain $X \to Z \in \Sigma^+_{\mathfrak{F}}$ by the decomposition rule. From $X \to Z \in \Sigma^+_{\mathfrak{F}}$, $Z \to Y \in \Sigma$ and $Z \subseteq XT_s$ we conclude $X \to Y \in \Sigma^+_{\mathfrak{F}}$ by an application of the null transitivity rule. That is, also after the $j + 1$st run we have $Closure \subseteq X^+_{\mathfrak{F}}$.

It remains to show that $X^+_{\mathfrak{F}} \subseteq Closure$ holds as well. For this purpose, consider the chain

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \ldots \subset \Sigma_k = \Sigma^+_{\mathfrak{F}}$$

where Σ_j results from Σ_{j-1} by application of a single inference rule from $\mathfrak{F}$, for $j = 1, \ldots, k$. We will use induction on j to show the following

$$\text{if } Z \to Y \in \Sigma_j \text{ and } Z \subseteq Closure \cap XT_s, \text{ then } Y \subseteq Closure.$$

Then we conclude for $j = k$ that $Y \subseteq Closure$ follows from $Z \to Y \in \Sigma^+_{\mathfrak{F}}$ and $Z \subseteq Closure \cap XT_s$. Hence, $X^+_{\mathfrak{F}} \subseteq Closure$ follows for $Z = X$ and $Y = X^+_{\mathfrak{F}}$.

We proceed by induction on j. If $j = 0$, then $Z \to Y \in \Sigma$. If $Z \subseteq Closure \cap XT_s$, then $\emptyset \to Y$ is in *FDList* due to line 5-7, and Y will be added to *Closure* in line 9. Hence, $Y \subseteq Closure$. If $j > 0$, then the FD $Z \to Y \in \Sigma_j - \Sigma_{j-1}$ has been inferred by application of one of the four inference rules in $\mathfrak{F}$. If $Z \to Y$ results from the reflexivity axiom, then $Y \subseteq Z \subseteq$ *Closure*, where $Z \subseteq$ *Closure* follows from the induction hypothesis. If $Z \to Y$ results from applying the decomposition rule to $Z \to YU \in \Sigma_{j-1}$, then $Y \subseteq YU \subseteq$ *Closure*, where $YU \subseteq$ *Closure* follows from the induction hypothesis. If $Z \to Y$ results from applying the union rule to $Z \to U \in \Sigma_{j-1}$ and $Z \to V \in \Sigma_{j-1}$, then $Y = UV \subseteq$ *Closure*, where $UV \subseteq$ *Closure* follows from the induction hypothesis. If $Z \to Y$ results from applying the null transitivity rule to $Z \to U \in \Sigma_{j-1}$ and $U \to Y \in \Sigma_{j-1}$, then $U \subseteq ZT_s$. In particular, if $Z \subseteq$ *Closure* $\cap XT_s$, then the hypothesis tells us that $U \subseteq$ *Closure* and $U \subseteq ZT_s \subseteq XT_s$. Applying the hypothesis to $U \to Y \in \Sigma_{j-1}$ and $U \subseteq$ *Closure* $\cap XT_s$ results in $Y \subseteq$ *Closure*. This concludes the proof.

We now establish the worst-case linear time complexity of deciding the implication problem for keys and FDs over table schemata with a null-free subschema. For that purpose, we first define the sizes of the measures we use. The size $|\varphi|$ of a key or FD φ is the total number of attributes occurring in φ, and the size $||\Sigma||$ of Σ is the sum of $|\sigma|$ over all elements $\sigma \in \Sigma$. Further, let

$$\Sigma[U] := \{V \to W \in \Sigma \mid V \subseteq U\} \cup \{uV \in \Sigma \mid V \subseteq U\}$$

define the *U-guard* of Σ.

Theorem 8 *The problem whether a key $\varphi = uX$ or FD $\varphi = X \to Y$ is implied by a set Σ of keys and FDs over table schema T with NFS nfs(T_s) can be decided in $\mathcal{O}(||\Sigma|| + ||\Sigma[XT_s] \cup \{\varphi\}||)$ time.*

Proof Corollary 2 shows how to reduce the implication problem to the computation of the attribute closure with respect to Σ[FD] and *nfs*(T_s). Algorithm 2 can be sped up by first computing the set $\Sigma[XT_s]$ by a single pass over Σ, given $\varphi = uX$ or $\varphi = X \to Y$, and *nfs*(T_s). The test for attributes to be in *Closure* $\cap XT_s$ of line 5 in Algorithm 2 then reduces to a test for attributes to be in *Closure*. Every attribute in $\Sigma[XT_s] \cup \{\varphi\}$ is then used at most once during the **repeat** loop.

Corollary 3 *Let $\Sigma \cup \{\varphi\}$ denote a set of SQL keys and FDs over table schema T with NFS nfs(T_s), where φ denotes either the key uX or FD $X \to Y$. Then the following are equivalent:*

1. $\Sigma \models_{T_s} X \to Y$ *if and only if* $\Sigma[XT_s][FD] \models X \to Y$
2. $\Sigma \models_{T_s} uX$ *if and only if* $\Sigma[XT_s][FD] \models X \to T$ *and* $uZ \in \Sigma[XT_s]$.

Proof The first equivalence follows from the correctness of Algorithm 2, and its equivalence to the standard computation of the attribute set closure $X^+_{\Sigma[XT_s][FD]}$ with respect to the FD set $\Sigma[XT_s][FD]$ that only contains FDs $V \to W$ where $V \subseteq XT_s$.

The second equivalence follows from the second equivalence of Corollary 2, and the first equivalence we have just established.

We conclude this section by analyzing the examples from the previous section from an algorithmic point of view.

Example 4 *Consider the relation schema* Work *with attributes emp, dpt and mng, NFS* $\text{Work}_s = \{\textit{dpt}, \textit{mng}\}$*, and*

$$\Sigma = \{\textit{emp} \to \textit{dpt}, \textit{dpt} \to \textit{mng}, u(\textit{mng})\}.$$

Then $\Sigma \models_{\text{Work}_s} u(\textit{emp})$*. Indeed, Algorithm 2 computes*

$$\{\textit{emp}\}^*_{\Sigma[FD],\text{Work}_s} = \{\textit{emp}, \textit{dpt}, \textit{mng}\}$$

and $u(\textit{mng}) \in \Sigma$ *where* $\{\textit{mng}\} \subseteq \{\textit{emp}\} \cup \text{Work}_s$*. Corollary 2 tells us therefore that* $\Sigma \models_{\text{Work}_s} u(\textit{emp})$*.* ∎

Example 5 *Consider the relation schema* Work *with attributes emp, dpt and mng, NFS* $\text{Work}_s = \{\textit{dpt}, \textit{mng}\}$*, and*

$$\Sigma = \{\textit{emp} \to \textit{dpt}, \textit{dpt} \to \textit{mng}\}.$$

Then $\Sigma \not\models_{\text{Work}_s} u(\textit{emp})$*. Indeed, Algorithm 2 computes*

$$\{u(\textit{emp})\}^*_{\Sigma[FD],\text{Work}_s} = \{\textit{emp}, \textit{dpt}, \textit{mng}\}$$

but there is no key in Σ*. Therefore, Corollary 2 tells us that* $\Sigma \not\models_{\text{Work}_s} u(\textit{emp})$*.* ∎

Example 6 *Consider the relation schema* Work *with attributes emp, dpt and mng, NFS* $\text{Work}_s = \{\textit{dpt}\}$*, and*

$$\Sigma = \{\textit{emp} \to \textit{dpt}, \textit{dpt} \to \textit{mng}, u(\textit{mng})\}.$$

Then $\Sigma \not\models_{\text{Work}_s} u(\textit{emp})$*. Indeed, Algorithm 2 computes*

$$\{\textit{emp}\}^*_{\Sigma[FD],\text{Work}_s} = \{\textit{emp}, \textit{dpt}, \textit{mng}\}$$

but for $u(\textit{mng}) \in \Sigma$ *we have* $\{\textit{mng}\} \not\subseteq \{\textit{emp}\} \cup \text{Work}_s$*. Therefore, Corollary 2 tells us that* $\Sigma \not\models_{\text{Work}_s} u(\textit{emp})$*.* ∎

B Possibilistic SQL Constraints

We will now describe how to reduce the implication problem of p-SQL constraints to the implication problem of traditional SQL constraints. Therefore, we define the cuts Σ_i of a set Σ of p-SQL constraints as the set of traditional SQL constraints that appear in Σ with c-degree β_i or higher.

Definition 3 *Let* Σ *denote a set of p-SQL keys, possibilistic functional dependencies and possibilistic* ***NOT NULL*** *constraints over a p-SQL table schema* $(T, \mathcal{S}_p)$*. For every c-degree* $\beta_i < \beta_{k+1}$ *we define the* β_i*-cut* Σ_i *as the set of SQL keys, functional dependencies and* ***NOT NULL*** *constraints* σ *over table schema* T *for which there is some p-constraint* $(\sigma, \beta) \in \Sigma$ *such that* $\beta \geq \beta_i$*, that is,*

$$\Sigma_i = \{\sigma \mid (\sigma, \beta) \in \Sigma \wedge \beta \geq \beta_i\}\ .$$

Consider the p-table schema $(\textsc{Work}, \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4)$ with the following set Σ of p-constraints: $(dpt \to mng, \beta_1)$, $(emp \to dpt, \beta_2)$, and $(u(emp), \beta_3)$. Then the β-cuts of Σ are: $\Sigma_1 = \{dpt \to mng\}$, $\Sigma_2 = \{dpt \to mng, emp \to dpt\}$, and $\Sigma_3 = \{dpt \to mng, emp \to dpt, u(emp)\}$. The importance of β-cuts results from the following theorem. Hence, we can decide instances $\Sigma \models (\varphi, \beta_i)$ of implication problems for p-SQL constraints by deciding instances $\Sigma_i \models \varphi$ of implication problems for traditional SQL constraints.

Theorem 9 *Let $\Sigma \cup \{(\varphi, \beta_i)\}$ denote a set of p-SQL keys, functional dependencies, and* `NOT NULL` *constraints over a p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$ where $1 \leq i \leq k$. Then $\Sigma \models (\varphi, \beta_i)$ if and only if $\Sigma_i \models \varphi$.*

Proof We first show that $\Sigma_i \models \varphi$ is sufficient for $\Sigma \models (\varphi, \beta_i)$ to hold. Let us assume that $\Sigma \models (\varphi, \beta_i)$ does not hold. We will show that $\Sigma_i \models \varphi$ does also not hold. Based on our assumption there is some p-table (t, Poss_t) that violates (φ, β_i) and satisfies all $(\sigma, \beta) \in \Sigma$. By definition, t_{k+1-i} violates φ, and for all $\beta_j \geq \beta_i$ it follows that t_{k+1-j} satisfies every $\sigma \in \Sigma_j$. From $\beta_j \geq \beta_i$ it follows $t_{k+1-i} \subseteq t_{k+1-j}$. Consequently, by soundness of the *submodel* rule we conclude that t_{k+1-i} violates φ, and t_{k+1-i} satisfies every $\sigma \in \Sigma_i$. We conclude that $\Sigma_i \models \varphi$ does not hold.

We now show that $\Sigma_i \models \varphi$ is necessary for $\Sigma \models (\varphi, \beta_i)$ to hold. Let us assume that $\Sigma_i \models \varphi$ does not hold. We will show that $\Sigma \models (\varphi, \beta_i)$ does also not hold. Based on our assumption there is some table t that violates φ and satisfies Σ_i. We will now construct a p-table $\bar{t}$ from t that violates (φ, β_i) and satisfies Σ. If φ is a `NOT NULL` constraint, then t contains some row r_1 that violates φ. Otherwise, t contains two rows r_0, r_1 that violate φ. If t contains at least two rows, then $\bar{t} := \{r_0, r_1\}$ and we assign to r_0 the p-degree α_1 and to r_1 the p-degree α_{k+1-i}. If $t = \{r_1\}$, then $\bar{t} := \{r_0, r_1\}$ for a new row r_0 with non-null domain values that do not occur in r_1 and assign the p-degree α_1 to r_0 and the p-degree α_{k+1-i} to r_1. The p-table $\bar{t}$ has the property that for every $\beta_k \leq \beta_j < \beta_i$, $\bar{t}_{k+1-j} = \{r_0\}$, and for every $\beta_j \geq \beta_i$, $\bar{t}_{k+1-i} = \{r_0, r_1\}$. Consequently, the p-table $(\bar{t}, \mathit{Poss}_{\bar{t}})$ satisfies Σ but violates (φ, β_i). That is, $\Sigma \models (\varphi, \beta_i)$ does not hold.

Consider the p-table schema $(\textsc{Work}, \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4)$ with the following set Σ of p-constraints: $(dpt \to mng, \beta_1)$, $(emp \to dpt, \beta_2)$, and $(u(emp), \beta_3)$. We have seen that Σ does not imply $(emp \to mng, \beta_2)$. Using Theorem 9 this is equivalent to noting that Σ_2 does not imply $emp \to mng$. We have also seen that if Θ results from Σ by adding the `NOT NULL` constraint $(n(dpt), \beta_3)$, then Θ still does not imply $(emp \to mng, \beta_2)$. Using Theorem 9 this is equivalent to noting that Θ_2 does not imply $emp \to mng$. This is shown by Table 8. Finally, we had seen that if Ω results from Σ by adding the `NOT NULL` constraint $(n(dpt), \beta_2)$, then Ω does imply $(emp \to mng, \beta_2)$. Using Theorem 9 this is equivalent to noting that Ω_2 does imply $emp \to mng$.

Table 8: SQL Example

emp	*dpt*	*mng*
Andy	$\perp$	Sofia
Andy	$\perp$	Sam

B.1 Axiomatic Characterization

Theorem 10 (Theorem 1 restated) *The set $\mathfrak{pS}$ forms a finite axiomatization for the implication problem of the combined class of p-SQL keys, FDs, and* `NOT NULL` *constraints.*

Proof We first show the soundness of the inference rules. The soundness of the *submodel* rule follows from the following: if the world t_{k+1-i} satisfies σ, then so does $t_{k-i} \subseteq t_{k+1-i}$. The soundness of the *triviality* axiom is a consequence of the definition: a constraint always holds with bottom c-degree. The soundness of the remaining rules follows from Theorem 9 and the soundness of the corresponding inference rules from the non-possibilistic SQL case, as proven in Lemma 1.

It remains to show the completeness of the inference rules. Here, we derive the completeness directly from the completeness of the inference rules in $\mathfrak{S}$ for the implication of SQL keys, FDs and `NOT NULL` constraints, and Theorem 9. Indeed, assume that we have $\Sigma \models (\varphi, \beta_i)$ for some arbitrarily given set $\Sigma \cup \{(\varphi, \beta_i)\}$ of p-constraints over p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$. If $\beta_i = \beta_{k+1}$, then we can infer (φ, β_i) directly by a single application of the *triviality* axiom. Otherwise, $\beta_i > \beta_{k+1}$. In this case, Theorem 9 shows that $\Sigma_i \models \varphi$ holds. By completeness of the inference rules $\mathfrak{S}$ from Table 7, we conclude that $\Sigma \vdash_{\mathfrak{S}} \varphi$. Now, we define the set $\Sigma_i^i = \{(\sigma, \beta_i) \mid \sigma \in \Sigma_i\}$. Next, we obtain the inference $\Sigma_i^i \vdash_{\mathfrak{pS}} (\varphi, \beta_i)$ from the inference $\Sigma \vdash_{\mathfrak{S}} \varphi$ by replacing each application of an inference rule in $\mathfrak{S}$ with an application of the corresponding inference rule in $\mathfrak{pS}$, simply by augmenting each constraint σ with the c-degree β_i and the condition $Y \subseteq XT_s$ with the condition $Y \subseteq XT_s^i$. From $\Sigma_i^i \vdash_{\mathfrak{pS}} (\varphi, \beta_i)$ we finally obtain $\Sigma \vdash_{\mathfrak{pS}} (\varphi, \beta_i)$ since for every $(\sigma, \beta_i) \in \Sigma_i^i$ there is some $(\sigma, \beta_j) \in \Sigma$ such that $\beta_j \geq \beta_i$ and we can successively apply the *submodel* rule to (σ, β_j) to derive (σ, β_i). This concludes the proof.

Consider the p-table schema $(\textsc{Work}, \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4)$ with the following set Σ of p-constraints: $(dpt \to mng, \beta_1)$, $(emp \to dpt, \beta_2)$, and $(u(emp), \beta_3)$. We have seen that $\Omega = \Sigma \cup \{(n(dpt), \beta_2)\}$ implies $(emp \to mng, \beta_2)$. Indeed, $\Omega_2 = \{dpt \to mng, emp \to dpt, n(dpt)\}$, and an application of *null FD transitivity* results in $\Omega_2 \vdash_{\mathfrak{S}} emp \to mng$.

B.2 Algorithmic Characterization

Theorem 11 (Theorem 2 restated) *Let Σ denote a set of p-SQL keys, FDs, and `NOT NULL` constraints over p-SQL table schema $(T, \alpha_1 > \cdots > \alpha_{k+1})$. For all $(X \to Y, \beta_i)$, (uX, β_i) and (nX, β_i) over $(T, \alpha_1 > \cdots > \alpha_{k+1})$ the following hold:*

1. *$\Sigma \models (X \to Y, \beta_i)$ if and only if $Y \subseteq X^+_{\Sigma_i[XT_s^i][FD]}$.*
2. *$\Sigma \models (uX, \beta_i)$ if and only if $X^+_{\Sigma_i[XT_s^i][FD]} = T$ and there is some $uZ \in \Sigma_i[XT_s^i]$.*
3. *$\Sigma \models (nX, \beta_i)$ if and only if $X \subseteq T_s^i$.*

Proof We show each of the properties in turn.

1. By Theorem 9, $\Sigma \models (X \to Y, \beta_i)$ holds if and only if $\Sigma_i \models_{T_s^i} X \to Y$. However, from the first equivalence of Corollary 3 it follows that $\Sigma \models (X \to Y, \beta_i)$ holds if and only if $\Sigma_i[XT_s^i][FD] \models X \to Y$ holds. It is a classical result [2] that $\Sigma_i[XT_s^i][FD] \models X \to Y$ holds if and only if $Y \subseteq X^+_{\Sigma_i[XT_s^i][FD]}$.

2. By Theorem 9, $\Sigma \models (uX, \beta_i)$ holds if and only if $\Sigma_i \models_{T_s^i} uX$ holds. However, from the second equivalence of Corollary 3 it follows that $\Sigma_i \models_{T_s^i} uX$ holds if and only if $\Sigma_i[XT_s^i][FD] \models X \to T$ holds and there is some $uZ \in \Sigma_i[XT_s^i]$. It is a classical result [2] that $\Sigma_i[XT_s^i][FD] \models X \to T$ holds if and only if $X^+_{\Sigma_i[XT_s^i][FD]} = T$.

3. By Theorem 9, $\Sigma \models (nX, \beta_i)$ holds if and only if $\Sigma_i \models nX$ holds. However, $\Sigma_i \models nX$ holds if and only if $X \subseteq T_s^i$ holds.

B.3 Logical Characterization

Consider the p-table schema (WORK, $\alpha_1 > \alpha_2 > \alpha_3 > \alpha_4$) with the following set Θ of p-constraints: $(dpt \to mng, \beta_1)$, $(emp \to dpt, \beta_2)$, $(u(emp), \beta_3)$, and $(n(dpt), \beta_3)$. We have seen that Θ does not imply the p-FD $(\varphi = emp \to mng, \beta_2)$. Indeed, $\Theta_2[emp] = \{emp \to dpt\}$, so the translation into Horn formulae would result in $(\Theta_2[emp])' = \{\neg emp' \vee dpt'\}$ and $\varphi' = \neg emp' \vee mng'$. Now the set

$$\{\neg emp' \vee dpt', emp', \neg mng'\}$$

of Horn formulae is satisfiable, confirming that Θ does not imply the p-FD $(emp \to mng, \beta_2)$.

Theorem 12 (Theorem 4 restated) *The implication problem of p-SQL keys, FDs, and* `NOT NULL` *constraints is* PTIME*-complete and equivalent to* HORNSAT*.* ▮

Proof Based on the first two equivalences in Theorem 2 and the soundness of *key weakening* in $\mathfrak{B}$ from Table 6, it follows that:

1. $\Sigma \models (X \to Y, \beta_i)$ if and only if $\Sigma_i[XT_s^i] \models X \to Y$, and

2. $\Sigma \models (uX, \beta_i)$ if and only if $\Sigma_i[XT_s^i] \models X \to T$ and there is some $uZ \in \Sigma_i[XT_s^i]$.

Note that $\Sigma_i[XT_s^i]$ is a set of keys and FDs over bag schemata. Indeed, the condition $\Sigma_i[XT_s^i] \models X \to T$ and there is some $uZ \in \Sigma_i[XT_s^i]$ is equivalent to $\Sigma_i[XT_s^i] \models uX$ by soundess of the *pullback* rule in $\mathfrak{B}$ from Table 6.

Hence, we have reduced instances $\Sigma \models (\sigma, \beta_i)$ for keys $\sigma = uX$ and FDs $\sigma = X \to Y$ of the implication problem for our class of p-constraints to instances $\Sigma_i[XT_s^i] \models \sigma$ for the implication problem of keys and FDs over bag schemata. For the latter it is known that keys behave like propositional goal clauses and FDs behave like propositional definite clauses [14].

In fact, it is well-known that the implication problem of Boolean propositional Horn formulae is PTIME-complete and equivalent to HORNSAT [8], the problem of deciding satisfiability for a set of propositional Boolean Horn formulae. The latter can be solved by using linear resolution.