Inclusion dependencies and their interaction with functional dependencies in SQL

Henning Koehler, Sebastian Link

PII: S0022-0000(16)30116-7
DOI: http://dx.doi.org/10.1016/j.jcss.2016.11.004
Reference: YJCSS 3037

To appear in: *Journal of Computer and System Sciences*

Received date: 17 December 2015
Revised date: 19 August 2016
Accepted date: 14 November 2016

Highlights

- Simple inclusion dependencies and NOT NULL constraints are not finitely axiomatizable.
- The new class of not null inclusion dependencies (NNINDs) subsumes simple and partial inclusion dependencies.
- The implication problem for NNINDs is finitely axiomatizable, PSPACE- complete, and fixed parameter-tractable in their arity.
- Typed acyclic NNINDs are NP-complete, and tree-like NNINDs are linear time decidable.
- Super-reducedness is an efficient condition sufficient to guarantee no interaction between functional dependencies and NNINDs.

Inclusion Dependencies and their Interaction with Functional Dependencies in SQL

Henning Koehler[a], Sebastian Link[b]

[a]*School of Engineering & Advanced Technology, Massey University, New Zealand*
[b]*Department of Computer Science, University of Auckland, New Zealand*

Abstract

Driven by the SQL standard, we investigate simple and partial inclusion dependencies (INDs) with not null constraints. Implication of simple INDs and not null constraints is not finitely axiomatizable. We propose not null inclusion dependencies (NNINDs) that subsume simple and partial INDs, are finitely axiomatizable and PSPACE-complete to decide. NNINDs are fixed parameter-tractable in their arity, typed acyclic NNINDs are NP-hard, and tree-like NNINDs are decidable in linear time. We use a chase to decide implication for functional dependencies and acyclic NNINDs in exponential time, and identify a liberal condition that guarantees no interaction between functional dependencies and acyclic (NN)INDs.

Keywords: Axiomatization, Chase, Complexity, Functional dependency, Inclusion dependency, Null, Partial semantics, Simple semantics, Undecidability, SQL

1. Introduction

Motivation. Domain, entity and referential integrity form the three most fundamental classes of integrity rules, already proposed by Codd in his seminal paper [20]. Referential integrity is enforced by foreign keys and, more generally, inclusion dependencies (INDs). INDs empower us to specify which data must be duplicated in what relations, they can express referential integrity at the logical level and are an invaluable tool for classical data management tasks, including conceptual data modeling [18], schema design [28, 42, 46], and query answering [14, 34, 51]. An IND $R[X] \subseteq S[Y]$ with attribute sequences $X = [A_1, \ldots, A_n]$

Email addresses: `h.koehler@massey.ac.nz` (Henning Koehler), `s.link@auckland.ac.nz` (Sebastian Link)

The submission is an extended version of our conference paper *Inclusion dependencies reloaded*, presented at the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015), Melbourne, Australia. The research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.

Corresponding author

and $Y = [B_1, \ldots, B_n]$ on relation schemata R and S, respectively, is satisfied by a relational database with R-relation r and S-relation s, if for every R-tuple t there is some S-tuple t' such that for all $i = 1, \ldots, n$, the value $t(A_i)$ matches the value $t'(B_i)$. An example IND in the TPC-C schema[4] is

$$\text{ORDER}[\text{c\_id, d\_id, w\_id}] \subseteq \text{CUSTOMER}[\text{c\_id, d\_id, w\_id}]$$

which states that every order references some customer c_id who placed the order in a district d_id of a warehouse w_id. As {c_id, d_id, w_id} forms a key over CUSTOMER, this IND is even a foreign key.

In the context of the relational model, INDs and their core computational problem of deciding implication have received dedicated interest, see for example [15, 21, 22, 40, 41, 49, 50]. The $\mathcal{C}$-implication problem for a class $\mathcal{C}$ of data dependencies is to decide whether for every given database schema and every given set $\Sigma \cup \{\varphi\}$ of dependencies from $\mathcal{C}$ over the schema, every database instance that satisfies all dependencies in Σ also satisfies φ.

The $\mathcal{C}$-implication problem has always been the central problem in logic [8], and has several important motivations in database theory and practice. For example, solutions to the implication problem allow database designers to decide whether some data dependency φ must be specified on top of the set Σ of dependencies that have already been specified on the given database schema in order to guarantee data integrity under updates. Similarly, they allow designers to compute sets Σ of data dependencies in which no element $\varphi \in \Sigma$ is enforced redundantly, thereby ensuring a minimal overhead in enforcing data integrity.

In schema normalization, such as 3NF synthesis or BCNF decomposition, it must be decided whether a sub-schema satisfies the given normal form condition, which is only achieved by deciding the associated implication problem [7, 26, 42]. In query optimization, deciding implication makes it possible to eliminate superfluous joins or `DISTINCT` clauses [1, 29], for example. In database security, the implication problem of data dependencies is central because implied data dependencies enable users to bypass access control and infer confidential information [6, 27]. Renewed interest in INDs comes from modern applications such as data cleaning and schema matching [9, 45], data integration [10], data profiling [4, 19, 56], as well as query answering and the semantic web [13, 29, 44, 52, 55].

Surprisingly, INDs have only received little attention in the context of the industry standard SQL, which subsumes null-free relations as the idealized special case where all attributes of the schema have been declared not null, that is, the null marker $\perp$ must not feature in any of the table columns. In fact, the SQL standard proposes a 'simple' and a 'partial' semantics for INDs. Both semantics apply the definition from the null-free relational case above, but they differ in which tuples t they consider. For an IND $R[X] \subseteq_p S[Y]$ under partial semantics, all tuples t are considered, and $t(A_i)$ matches $t'(B_i)$ if and only if $t(A_i) = \perp$ or $t(A_i) = t'(B_i)$ holds. For an IND $R[X] \subseteq_s S[Y]$ under simple semantics there is a distinction between R-tuples t which are total on X, that

[4]`http://www.tpc.org/tpcc/`

is, where for all $i = 1, \dots, n$, the value $t(A_i)$ is different from $\perp$, and R-tuples t which are partial on X, that is, not total on X. Only X-total tuples t are considered, and $t(A_i)$ matches $t'(B_i)$ if and only if $t(A_i) = t'(B_i)$ holds. We shall refer to INDs under simple (partial) semantics as simple (partial) INDs.

For an example, consider a database where the only tuple t of the database occurs in the ORDER-relation and has projection $[\perp, d_1, w_1]$ on $[c\_id, d\_id, w\_id]$. This database satisfies the simple IND

$$\text{ORDER}[\text{c\_id, d\_id, w\_id}] \subseteq_s \text{CUSTOMER}[\text{c\_id, d\_id, w\_id}] \ ,$$

but violates the partial IND

$$\text{ORDER}[\text{c\_id, d\_id, w\_id}] \subseteq_p \text{CUSTOMER}[\text{c\_id, d\_id, w\_id}] \ .$$

It is well-known that simple semantics for foreign keys is natively supported by all SQL implementations, but partial semantics is not natively supported by any SQL implementation [31, 48].

On the one hand, it is not surprising that the implication problems for INDs under simple and partial semantics are different from one another. For example, the partial IND above implies the partial IND

$$\text{ORDER}[\text{d\_id, w\_id}] \subseteq_p \text{CUSTOMER}[\text{d\_id, w\_id}] \ ,$$

but the simple IND above does not imply the simple IND

$$\text{ORDER}[\text{d\_id, w\_id}] \subseteq_s \text{CUSTOMER}[\text{d\_id, w\_id}] \ .$$

On the other hand, it is surprising that the implication problem has only been investigated under partial semantics, and only under the assumption that the null marker can occur in every column [38]. In fact, we will see that the implication problem for INDs under partial semantics and not null constraints enjoys the same axiomatization as INDs on null-free relational databases [15]. In addition, for simple INDs alone (i.e., without not null constraints) we will establish an axiomatization that results from the axiomatization for partial INDs by omitting the projection rule.

However, things already change dramatically when we consider simple INDs in combination with not null constraints. In this case, we will show that the associated implication problem is not k-ary axiomatizable for any finite k, i.e., cannot be axiomatized by inference rules with at most k premises, and in particular not by any finite set of rules. This result is rather discouraging for database practice in which simple semantics is natively supported, but not partial semantics [31, 48]. Concise axiomatizations aid in human understanding of constraint interaction, and critical applications such as schema design, integrity enforcement and query processing. We emphasize the importance of the ability to reason about simple and partial INDs under not null constraints by the following example from database design.

Example 1. *Consider the following relation schemata, in which attributes A that do not permit null values are denoted by A[NN]:*

- CONFERENCE*(CName[NN])*,
- PARTICIPANT*(PName[NN])*,
- CHAIR*(CName[NN],PName[NN])*,
- BOOKING*(CName,PName,Dates[NN],Room[NN]), and*
- TALK*(CName,PName,Title[NN]).*

The designer has identified the following inclusion dependencies:

- CHAIR*[CName]* $\subseteq_p$ CONFERENCE*[CName]*,
- CHAIR*[PName]* $\subseteq_p$ PARTICIPANT*[PName]*,
- BOOKING*[CName, PName]* $\subseteq_s$ CHAIR*[CName, PName]*,
- TALK*[CName, PName]* $\subseteq_p$ BOOKING*[CName, PName]*.

If we were able to reason about simple and partial inclusion dependencies under not null constraints, this would allow us to detect the following:

1. *The IND* TALK*[CName, PName]* $\subseteq_s$ CHAIR*[CName, PName] is implied by the specified INDs, indicating that talks (with no null occurrences in CName and PName) can only be given by conference chairs, which clearly indicates a design flaw, and*

2. *The IND* TALK*[CName]* $\subseteq_s$ CONFERENCE*[CName] is not implied by the specified constraints, but should hold.*

The challenge. For the reasons outlined before, database research is expected to answer the fundamental question how the semantics of INDs should be defined in the context of the industry standard SQL. Ideally, the semantics can be defined in such a way that all of the following targets are met: i) simple and partial semantics are subsumed by the semantics as special cases, and ii) the semantics enjoys similar computational properties to those of its idealized special case in which all attributes are specified not null. More precisely, condition ii) means that the associated implication problem is finitely axiomatizable and PSPACE-complete to decide [15]. Ideally, there would be some expressive fragments whose associated implication problems are easier, potentially even tractable, to decide.

As entity integrity is primarily enforced by keys, and more generally functional dependencies (FDs), it is also important to identify decidable instances of the implication problem for the combined class of FDs and INDs. The challenge here is that each of the finite and unrestricted implication problems for the general combined class of functional and inclusion dependencies is already undecidable in the special case of null-free relational databases [17, 49].

Contributions. Strongly motivated by our observations about the simple and partial semantics of SQL inclusion dependencies, we address the challenges above as follows.

1. We investigate the implication problem of INDs under both simple and partial semantics, each in combination with not null constraints, as suggested by the SQL standard. The idealized special case where all attributes are declared

not null yields in each case the well-known semantics [15] of INDs over null-free relations.
2. In the presence of not null constraints, we show that partial semantics enjoys the same finite axiomatization as traditional INDs, but simple semantics is not k-ary axiomatizable for any finite k. The latter result already holds in the case where all relation schemata have at most four attributes.
3. We propose the class of "not null inclusion dependencies" (NNINDs), which exhibits a natural semantics, subsumes simple and partial semantics as special cases, enjoys a finite axiomatization and is PSPACE-complete to decide, thereby recovering the positive results about INDs in null-free relational databases under the more general framework of SQL-like databases.
4. Exploiting our completeness argument of our finite axiomatization, we establish a chase procedure for NNINDs, subsuming the well-known chase for INDs over null-free relations as an idealized special case.
5. We show that the arity of NNINDs is a parameter that, when fixed, makes their implication problem tractable. In contrast to typed INDs over null-free relations, implication of typed NNINDs is NP-hard and thus unlikely to be decidable in PTIME, already in the acyclic case.
6. We investigate the interaction of NNINDs with functional dependencies (FDs). As (finite) implication is undecidable for this combined class [49, 17], we establish a liberal, sufficient condition that guarantees the non-interaction [40, 47] of FDs and an acyclic set of (not null) inclusion dependencies. Non-interaction is very desirable in database practice, as the combined implication problem reduces to separate implication problems which may each be solved more easily in isolation. Our sufficient condition is liberal in the sense that it subsumes two existing sufficient conditions [40, 47] as special cases, and is also effective as it can be decided in quadratic time. Our non-interaction result follows from a chase procedure that can be used to decide the implication problem for FDs and any acyclic set of NNINDs. Finite and unrestricted implication coincide in this case and can be decided in exponential time.
Organization. We summarize related work in Section 2 to differentiate our contributions. In Section 3 we formally introduce simple and partial semantics for INDs. The axiomatization and non-axiomatizability, respectively, of partial and simple INDs and not null constraints is discussed in Section 4. NNINDs are introduced in Section 5, and their implication problem is axiomatized and shown to be PSPACE-complete. The chase and important fragments of NNINDs are identified in Section 6. Section 7 investigates the interaction of NNINDs and FDs. Some new results for the null-free relational model are discussed in Section 8. We conclude in Section 9 where we also comment on future work.

2. Related Work

Inclusion dependencies form one of the most prolific concepts in database research and practice. We cannot list all the contributions made possible by them. Some insight into their diverse application areas was already provided in the introduction. Instead, we will focus on the main achievements regarding the implication problem of inclusion dependencies and highlight the novelty of our contributions in light of these achievements.

Inclusion dependencies generalize the notion of referential integrity, which was known to the broader database community during the 1970s [24]. A seminal paper on inclusion dependencies is [15], in which inference rules are presented, a Chase procedure is established, and the PSPACE-completeness of the implication problem is shown. Acyclic inclusion dependencies were introduced in [54], and it was shown in [21] that their implication problem is NP-complete. For typed inclusion dependencies, i.e., inclusion dependencies of the form $R[X] \subseteq S[X]$ were each attribute is mapped to an attribute of the same name, such as those found in Example 1, implication is decidable in PTIME [16]. All of these results assume null-free relations.

Inclusion dependencies in the presence of null markers have been studied in [38], but only for partial semantics, and without not null constraints. For this case the authors showed that the associated implication problem enjoys the same axiomatization as inclusion dependencies over null-free relations, and is PSPACE-complete to decide. Indexes for enforcing partial inclusion dependencies have been investigated in [48].

We conclude that previous research has not investigated the important combined class of inclusion dependencies and not null constraints, neither under simple nor partial semantics as recommended by the SQL standard. Only the special cases in which either all or no attributes may contain null values have been studied, and the former only for partial semantics. Our result that the combined class of simple inclusion dependencies and not null constraints is not k-ary axiomatizable for any finite k is surprising and discouraging for current database practice. It is surprising since: i) in both idealized special cases inclusion dependencies are axiomatizable, and ii) under the 'no information' semantics of null markers [43], previous research [2, 33] has not seen cases in which adding not null constraints to other axiomatizable classes of equality- or tuple-generating constraints results in their non-axiomatizability. The result is discouraging for current database practice, because implementations of database systems natively support simple, but not partial semantics of INDs [31, 48].

In [39] it is shown that for functional dependencies under possible world semantics, no k-ary axiomatization exists. Possible world semantics captures only the null marker interpretation 'value exists but unknown' but not 'value does not exist'. For this reason we assume the popular and SQL-compliant 'no information' interpretation [3, 32, 33, 43], under which both partial and simple semantics are sensible and their study justified.

A fundamental result for null-free relational databases concerns the undecidability of each finite and unrestricted implication for the combined class of functional and inclusion dependencies [17, 49]. It has also been shown that neither the finite nor the unrestricted implication problem for the combined class of functional and inclusion dependencies is k-ary axiomatizable for any k, in particular no finite axiomatization exists [15]. For the combined class of functional and acyclic inclusion dependencies, finite and unrestricted implication coincide, and require exponential exponential time to decide [21, 23]. Under the assumption that all attributes permit null marker occurrences, it was shown [38] that the implication problem for the combined class of ‘no information’ functional dependencies and inclusion dependencies under partial semantics is EXPTIME-complete to decide.

Based on the infeasibility of the general combined implication problem of FDs and INDs, several sufficient conditions on the structure of the given constraint sets have been identified that guarantee no interaction between these fragments of the constraint classes. These include unary inclusion dependencies [47, Theorem 10.20] and key-based inclusion dependencies on schemata in Boyce-Codd normal form [47, Theorem 10.21]. In [12] the notion of non-key-conflicting IND is introduced and used to establish a non-interaction result between INDs and keys. Our sufficient condition extends the condition of [12] to NNINDs and FDs over relations containing nulls and not null constraints, and is strictly more liberal than the conditions in [47], capturing more cases that can be handled efficiently. Levene and Loizou [40, 41] consider a stricter definition of ‘no interaction’, requiring that no subsets of the given constraint sets interact. Our sufficient condition is closed under subsets and thus also applies to this stricter requirement.

The results related to the individual class of (NN)INDs have been announced in [35]. The current submission is an extension of [35] in which we present all proofs, additional motivation and many examples that illustrate our concepts and results. In addition, the content of Section 7 is new, that is, all results related to the combined class of FDs and NNINDs. These include a chase procedure for FDs and NNINDs, which can be used to decide implication for any set of FDs and acyclic NNINDs in exponential time. In particular, the termination of the chase means that the finite and unrestricted implication problems coincide for FDs and acyclic NNINDs. In addition, we establish a sufficient condition for the non-interaction of FDs and acyclic NNINDs. The condition can be decided in quadratic time, and subsumes different sufficient conditions from the literature [40, 41] as special cases.

3. Inclusion dependencies in SQL

In this section we give preliminary definitions and formalize the notions of partial and simple inclusion dependencies in the context of SQL-like databases.

Let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a countably infinite set of symbols, called *attributes* or *columns*. A *relation schema* is a finite non-empty sequence $R = [A_1, \ldots, A_n]$ of distinct attributes in $\mathfrak{A}$. *Database schemata* are finite sets of relation schemata. Each attribute A of a relation schema R is associated with a countably infinite domain $dom(A)$ of the possible values that can occur in column A. To encompass partial information the domain of every attribute contains a distinguished null marker, denoted by $\perp$. Note that the null marker $\perp$ is different from a domain value. The inclusion of $\perp$ into the domain is purely a syntactic convenience.

A *tuple* t over $R = [A_1, \ldots, A_n]$ (R-tuple or simply tuple, if R is understood) is an element of the Cartesian product $dom(A_1) \times \cdots \times dom(A_n)$. A *relation* r over R (R-relation or simply relation, if R is understood) is a set of R-tuples, possibly infinite. While SQL tables may contain duplicates, these have no impact on whether an IND (or later an FD, as considered in Section 7) holds, so it suffices to study relations. Sometimes, we also write *instance* or *table* when referring to a relation. We may also refer to a relation with null marker occurrences when we want to emphasize the fact that null markers can be present. We talk of *null-free relations* when considering the classic case where null markers are not permitted. When necessary we say that relations are finite.

A *database* assigns a relation to each of its relation schemata that are part of the underlying database schema. For $X = [A_{i_1}, \ldots, A_{i_m}]$ we use $t[X]$ ($r[X]$) to denote the *projection* of tuple t (relation r) on the attributes $A_{i_1}, \ldots, A_{i_m}$. That is, $r[X] = \{t[X] \mid t \in r\}$. For an R-tuple t (R-relation r) we say that t (r) is X-total if for all $A_{i_j} \in X$, $t[A_{i_j}] \neq \perp$ (all tuples in r are X-total).

SQL permits the specification of attributes as not null. We permit this as well, and denote the not null attributes of a relation schema R by NN(R), and the attributes on which a tuple t is different from $\perp$ by NN(t). For relations with null markers, different extensions of inclusion dependencies over null-free relations exist. We study the two most prominent extensions, called partial and simple semantics, as defined by the SQL standard.

Definition 1 (less informative tuple)**.**
Consider tuples t_1, t_2 over $R_1 = [A_1, \ldots, A_m]$ and $R_2 = [B_1, \ldots, B_m]$. We say that t_1 is less informative[5] than *t_2 (or t_2* is more informative than *t_1), written as $t_1 \sqsubseteq t_2$, if and only if*

$$t_1[A_i] \in \{ \perp, \ t_2[B_i] \} \qquad \textit{for all } i = 1, \ldots, m \,.$$

We write $r_1 \sqsubseteq r_2$ to indicate that for every tuple in relation r_1 there exists some more informative tuple in relation r_2.

For an illustration of less informative tuples and relations, consider an example from an Australian tourism company [30].

Example 2. *Tours in the* Tour *table have a tour_id, for example a tour such as the "Gold Coast Grand Tour" has tour_id GCG. Tours have a fixed set of sites they visit. Consider the following database instance:*

Booking

Visitor_id	***Tour_id***	***Site_code***	***Date***
1006	*BRF*	$\perp$	*Sep* 19th
1001	*BRT*	*OR*	*Nov* 21st
1008	$\perp$	*BB*	*Sep* 5th
1012	$\perp$	*MV*	*Nov* 2nd
1011	*RF*	$\perp$	*Oct* 5th

Tour

Tour_id	***Site_code***	***Site_name***
GCG	*OR*	*O'Reilly's*
BRT	*OR*	*O'Reilly's*
BRT	*MV*	*Movie World*
RF	*BB*	*Binna Burra*
RF	*OR*	*O'Reilly's*

[5]While "less or equally informative" would be more accurate, "less informative" is predominant in the literature.

Let r_1 denote the projection of Booking *onto* $\{$*tour_id, side_code*$\}$ *and let* r_2 *denote the projection of* Tour *onto* $\{$*tour_id, side_code*$\}$*. Then* $r_1 \sqsubseteq r_2$ *does not hold, since there is no* r_2*-tuple that is more informative than the* r_1*-tuple (BRF,*$\perp$*). If we remove the first tuple in* r_1 *then* $r_1 \sqsubseteq r_2$ *holds.*

We will now introduce the simple and partial semantics for inclusion dependencies over relations with null marker occurrences. These naturally extend the simple and partial semantics, respectively, which are recommended for foreign keys in the SQL standard.

Definition 2 (partial/simple IND)**.**
Let X and Y be attribute sequences without repeating attributes over relation schemata R and S. We call an expression of the form $R[X] \subseteq_p S[Y]$ a partial inclusion dependency*, and an expression of the form $R[X] \subseteq_s S[Y]$ a* simple inclusion dependency*. A partial IND $R[X] \subseteq_p S[Y]$ holds for tables r, s over R, S if and only if $r[X] \sqsubseteq s[Y]$. A simple IND $R[X] \subseteq_s S[Y]$ holds for tables r, s over R, S if and only if for every X-total tuple $t_r \in r$ there exists a tuple $t_s \in s$ such that $t_r[X] = t_s[Y]$.*

Note that X, Y may be empty. In this case $t[\ ]$ is the empty tuple, and consequently $R[\ ] \subseteq_p S[\ ]$ and $R[\ ] \subseteq_s S[\ ]$ are equivalent and express that s must be non-empty if r is non-empty.

While only partial INDs have been considered in the literature [38], both partial and simple semantics are supported in the SQL standard (for foreign keys), and DBMSs support only simple semantics (and again only for foreign keys) [31, 48]. This motivates a closer investigation of simple INDs, and in particular their relationship to partial INDs.

Example 3. *Consider again the use case from Example 2. The simple inclusion dependency* Booking$[$*tour_id, site_code*$] \subseteq_s$ Tour$[$*tour_id, site_code*$]$ *is satisfied by the given database: The only total foreign key value (BRT,OR) in the* Booking *table is matched in the* Tour *table. However, the partial inclusion dependency* Booking$[$*tour_id, site_code*$] \subseteq_p$ Tour$[$*tour_id, site_code*$]$ *is violated by the given database: There is no* r_2*-tuple that is more informative than the* r_1*-tuple (BRF,*$\perp$*).*

The core computational problem in the theory of database constraints is the implication problem. It has several motivations in practice, inclusive of i) the ability to compute small representation systems (covers) for the sets of constraints that are actively enforced in a database system results in time savings that increase proportionally with the volume of the data that is being updated, and ii) the ability to decide whether a constraint is implied by a given set empowers database systems to decide whether a constraint can be used for optimizing a query on the fly.

For a given class $\mathcal{C}$ of constraints, the *(finite) implication problem* for $\mathcal{C}$ is to decide for a given finite set $\Sigma \cup \{\varphi\}$ of constraints in $\mathcal{C}$, whether Σ *(finitely) implies* φ. That is, whether every (finite) database that satisfies all the constraints in Σ also satisfies φ. Two sets Σ, Σ' of constraints are *semantically equivalent*, denoted by $\Sigma \equiv \Sigma'$, if either of the sets (finitely) implies all of the other set's constraints. For readability we will omit set brackets if Σ or Σ' are singletons.

Whether or not finite and unrestricted implication coincide depends on the class of constraints considered. E.g. for null-free relations it is know that they coincide for both the class of functional dependencies and the class of inclusion dependencies, but not for their combined class [49, 17].

Example 4. *We give some more details for Example 1. For that purpose let Σ denote the set of four inclusion dependencies from that example, and let φ denote the IND* Talk*[CName]* $\subseteq_s$ Conference*[CName]. We claimed in Example 1 that Σ does not imply φ. This can easily be witnessed by the database instance whose only tuple in the* Talk*-relation has projection (SIGMOD,$\perp$) on $\{$CName,PName$\}$, whose only tuple in the* Booking*-relation has projection (SIGMOD,$\perp$) on $\{$CName,PName$\}$, and whose other relations are empty. Clearly, all INDs are satisfied, but φ is violated because the CName SIGMOD does not appear in the* Conference*-relation.*

When talking about inclusion dependencies, we will need to identify subsets of the left-hand-side that "match" subsets of the right-hand-side. The following concept helps with this.

Definition 3 (induced mapping)**.**
Let $X = [A_1, \ldots, A_m]$ and $Y = [B_1, \ldots, B_m]$ be sequences of equal length, with distinct A_i. The mapping induced by X and Y, *denoted $(X \mapsto Y)$, is*

$$(X \mapsto Y)(A_i) = B_i$$

When applied to a set of values U, we obtain the images of values in $U \cap X$:

$$(X \mapsto Y)(U) = \{B_i \mid A_i \in U\}$$

When applied to a sequence of values U, we obtain a sequence of images instead.

We can now express an important relationship between partial and simple INDs precisely.

Theorem 1. *Every partial IND is semantically equivalent to a set of simple INDs, for finite as well as unrestricted implication. Specifically:*

$$R[X] \subseteq_p S[Y] \quad \equiv \quad \{\ R[X'] \subseteq_s S[Y'] \ \mid\ X' \subseteq X,\ Y' = (X \mapsto Y)(X')\ \}$$

Proof. It is easy to see that $R[X] \subseteq_p S[Y]$ implies every $R[X'] \subseteq_s S[Y']$ in the set.

To show the reverse, let r, s be instances of R, S so that every $R[X'] \subseteq_s S[Y']$ holds on r, s, and $t \in r$. From $R[X'] \subseteq_s S[Y']$ with $X' = \mathrm{NN}(t) \cap X$, it follows that there exists $t' \in s$ with $t[X'] = t'[Y']$. But this means $t[X] \sqsubseteq t'[Y]$, and since such a t' exists for all $t \in r$, $R[X] \subseteq_p S[Y]$ is satisfied. $\square$

Example 5. *The partial inclusion dependency*

$$\varphi : \textsc{Booking}[tour\_id, site\_code] \subseteq_p \textsc{Tour}[tour\_id, site\_code]$$

is semantically equivalent to the following set of four simple INDs:

- σ_1*:* $\textsc{Booking}[tour\_id, site\_code] \subseteq_s \textsc{Tour}[tour\_id, site\_code]$*,*
- σ_2*:* $\textsc{Booking}[tour\_id] \subseteq_s \textsc{Tour}[tour\_id]$*,*
- σ_3*:* $\textsc{Booking}[site\_code] \subseteq_s \textsc{Tour}[site\_code]$*, and*
- σ_4*:* $\textsc{Booking}[\,] \subseteq_s \textsc{Tour}[\,]$.

Note that σ_4*, which expresses that* Tour *is non-empty if* Booking *is non-empty, is needed here: a nearly empty database instance, whose only tuple occurs in the* Booking*-relation and has projection* $(\perp,\perp)$ *on* $\{tour\_id, site\_code\}$*, violates* φ *and* σ_4 *but satisfies* σ_1*,* σ_2*, and* σ_3*.*

Theorem 1 means that any DBMS which supports simple INDs indirectly supports partial INDs as well. Note that the opposite does not hold, i.e., simple INDs cannot be expressed by a set of partial INDs:

Example 6. *Consider the schemata* $R = A$*,* $S = B$ *with instances* $r = \{(\perp)\}$ *and* $s = \emptyset$*. Then the simple IND* $\varphi = R[A] \subseteq_s S[B]$ *holds on* r, s*, but the partial INDs* $R[\,] \subseteq_p S[\,]$ *and* $R[A] \subseteq_p S[B]$ *are violated. Similarly,* φ *holds for the instance* $r = \emptyset$ *and* $s = \{(\perp)\}$*, where the partial INDs* $S[\,] \subseteq_p R[\,]$ *and* $S[B] \subseteq_p R[A]$ *are violated. Thus* φ *implies no non-trivial partial INDs.*

Nevertheless, partial INDs are valuable in that they allow constraints to be expressed efficiently, i.e., without resorting to an exponential number of equivalent simple INDs.

4. Axiomatization

As mentioned earlier, concise axiomatizations aid in human understanding of constraint interaction, with critical applications in algorithm and schema design, constraint enforcement, query optimization, theorem proving and more. For example, while the connection between simple and partial inclusion dependencies, as described in Theorem 1, may appear obvious in retrospect, we only discovered it while searching for an axiomatization of NNINDs.

In this section, we will discuss axiomatizations for implication problems of partial and simple INDs, both in isolation and combination with not null constraints. Unless explicitly stated otherwise, we consider the *finite* implication problem. However, for all types of inclusion dependencies discussed here, taken by themselves (i.e., not in combination with FDs), the finite and unrestricted implication problems coincide. This will be easy to deduct from the axiomatizations we shall establish: they are complete for finite implication and hence for unrestricted implication as well, so one only needs to verify that each inference rule is still sound when relations can be infinite.

The notions of inference rules, soundness, completeness and (finite) axiomatization are standard [1]. We begin with a brief discussion of how not null constraints fit into our axiom systems.

4.1. Not null constraints

We treat not null constraints as properties of attributes, rather than separate integrity constraints. Essentially this means that we do not concern ourselves with the implication of not null constraints. This simplifies things, but raises the question of what would happen if we did consider their implication.

Not null constraints by themselves are pretty boring. If we express them as $\mathrm{NNC}(X)$ for non-empty attribute sets X, they can be axiomatized by subset and union rules $\mathrm{NNC}(XY) \vdash \mathrm{NNC}(X)$ and $\mathrm{NNC}(X), \mathrm{NNC}(Y) \vdash \mathrm{NNC}(XY)$.

The more interesting question is whether NNCs together with other classes of constraints can imply new NNCs, that are not implied by the given NNCs alone. The following lemmas and example provide some answers to this. Here INDs may be partial or simple, or even NNINDs as introduced in Section 5.

Lemma 1. *Let $\Sigma_{\texttt{NNC}}$ and $\Sigma_{\texttt{IND}}$ be sets of NNCs and INDs, respectively, and let σ be a NNC implied by $\Sigma_{\texttt{NNC}} \cup \Sigma_{\texttt{IND}}$. Then $\Sigma_{\texttt{NNC}}$ implies σ.*

Proof (sketch). Denote by $\mathrm{NN}(R)$ the set of all attributes in relation schema R appearing in $\Sigma_{\texttt{NNC}}$. For each relation schema R construct a relation r over R consisting of two tuples t_R, s_R defined as follows:

$$t_R[A] = \begin{cases} 0 & \text{if } A \in \mathrm{NN}(R) \\ \perp & \text{otherwise} \end{cases}$$
$$s_R = (0, \ldots, 0)$$

Then these relations form a database which satisfies $\Sigma_{\texttt{NNC}}$ and $\Sigma_{\texttt{IND}}$, but violates all not null constraints not implied by $\Sigma_{\texttt{IND}}$ alone. □

Similarly, FDs and acyclic INDs (see Section 7) taken together do not aid in the implication of NNCs. Our proof uses NNINDs.

Lemma 2. *Let $\Sigma_{\texttt{NNC}}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}}$ respectively be sets of NNCs, FDs and acyclic INDs, and let σ be a NNC implied by $\Sigma_{\texttt{NNC}} \cup \Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}}$. Then $\Sigma_{\texttt{NNC}}$ implies σ.*

Proof (sketch). Denote by R the relation schema to which σ applies. Consider the initial database $\mathbf{r}$ which consists of a single tuple t_R over R with

$$t_R[A] = \begin{cases} 0 & \text{if } A \in \mathrm{NN}(R) \\ \perp & \text{otherwise} \end{cases}$$

Clearly $\mathbf{r}$ violates every NNC on R not implied by $\Sigma_{\texttt{NNC}}$, but it may also violate $\Sigma_{\texttt{IND}}$. To fix this, we apply the chase (Section 7, Algorithm 2) to $(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$. As $\Sigma_{\texttt{IND}}$ is acyclic, the chase adds no tuples over schema R, so the `FD`-rule is never applied on R and t_R is not modified. Thus the resulting database still violates every NNC on R not implied by $\Sigma_{\texttt{NNC}}$, and satisfies $\Sigma_{\texttt{NNC}} \cup \Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}}$ by Theorem 13. □

We conclude our discussion of NNCs with an example showing that cyclic INDs together with FDs *can* help to imply new NNCs. Note however that the (finite) implication problem for INDs and FDs together is already undecidable for null-free relations [49, 17], and thus cannot be axiomatized anyway.

Example 7. *Consider the relation schema $R = AB$ with constraints*

$$\Sigma_{NNC} = \{ \text{NNC}(A) \} \quad \Sigma_{FD} = \{ A \to B \} \quad \Sigma_{IND} = \{ R[AB] \subseteq_p R[BA] \}$$

To show that these imply $\text{NNC}(B)$, assume that relation r over R satisfying these constraints violates $\text{NNC}(B)$, and thus contains a tuple $t_0 = (v_0, \perp)$. Then $R[AB] \subseteq_p R[BA]$ ensures that r contains $t_1 = (v_1, v_0)$, for some value v_1, and by repeated application there must exist $t_2 = (v_0, v_1) \in r$. From $\text{NNC}(A)$ it follows that v_0, v_1 are not null, so t_0, t_2 violate $A \to B$.

4.2. Finitely axiomatizable cases

We begin by establishing axiom systems for partial inclusions dependencies under not null constraints, and for simple inclusion dependencies without not null constraints.

The following definition falls in the same category as "induced mapping", in that it aids in formalizing what it means to rearrange the attributes on both sides of an IND "correspondingly".

Definition 4 (Index Permutation & Projection)**.** *An* index permutation & projection function of order (k,m)*, or (k, m)-IPP for short, with $k \leq m$, is an injective mapping $\sigma : \{1 \ldots k\} \to \{1 \ldots m\}$. We apply a (k, m)-IPP σ to a list L of length m by treating lists as position $\to$ value mappings, so that IPP application reduces to function composition $L \circ \sigma$:*

$$\sigma[A_1, \ldots, A_m] = [A_{\sigma(1)}, \ldots, A_{\sigma(k)}]$$

Where the order (k, m) of an IPP is clear from the context, we will not mention it explicitly. Given two lists X, Y, we denote the existence of an IPP function mapping X to Y by $X \subseteq_{IPP} Y$.

Example 8. *Let X denote the list [c_id, d_id, w_id] and Y denote the list [w_id, d_id]. Then $Y = \sigma(X)$ for the (3,2)-IPP $\sigma : \{1, 2, 3\} \to \{1, 2\}$ with $3 \mapsto 1$ and $2 \mapsto 2$.*

An axiomatization for partial INDs has been established by Levene and Loizou in [38, Theorem 3.3], and is identical to the axiom system for INDs on null-free relations previously established in [15]. This easily translates into an axiomatization which includes not null constraints.

Theorem 2. *The following axioms are sound and complete for partial inclusion dependencies on relations with not null constraints.*

$$\begin{array}{lll}
\textit{Reflexivity (R):} & \dfrac{}{R[X] \subseteq_p R[X]} & \\[2ex]
\textit{Transitivity (T):} & \dfrac{R[X] \subseteq_p S[Y] \quad S[Y] \subseteq_p T[Z]}{R[X] \subseteq_p T[Z]} & \\[2ex]
\textit{Projection \& Permutation (P):} & \dfrac{R[X] \subseteq_p S[Y]}{R[\sigma X] \subseteq_p S[\sigma Y]} & \sigma \textit{ is an IPP}
\end{array}$$

Proof. Soundness can easily be verified for each rule, but also follows as a consequence of their soundness in the absence of not null constraints [38, Theorem 3.3], as every database instance on a schema with not null constraints is still an instance over a schema where these constraints are omitted.

Completeness follows from [15, Theorem 3.1] which shows their completeness in relational databases without null values. Whenever an IND cannot be derived using the axioms, the theorem guarantees the existence of a counter-example showing non-implication. This counter-example contains no null values, and is still valid over a schema where null values are permitted for some columns. □

Example 9. *The partial IND*

$$\textsc{Order}[c\_id,\ d\_id,\ w\_id] \subseteq_p \textsc{Customer}[c\_id,\ d\_id,\ w\_id]$$

implies the partial IND

$$\textsc{Order}[w\_id,\ d\_id] \subseteq_p \textsc{Customer}[w\_id,\ d\_id]$$

which results from the (3,2)-IPP defined in Example 8. Indeed, the implication follows from the soundness of the Projection & Permutation rule for partial INDs.

As long as we do not consider not null constraints, the axiomatization of simple INDs is even simpler:

Theorem 3. *The following axioms are sound and complete for simple inclusion dependencies without not null constraints.*

$$\begin{array}{lll}
\textit{Reflexivity (R):} & \dfrac{}{R[X] \subseteq_s R[X]} & \\[2ex]
\textit{Transitivity (T):} & \dfrac{R[X] \subseteq_s S[Y] \quad S[Y] \subseteq_s T[Z]}{R[X] \subseteq_s T[Z]} & \\[2ex]
\textit{Permutation (P):} & \dfrac{R[X] \subseteq_s S[Y]}{R[\sigma X] \subseteq_s S[\sigma Y]} & \sigma \textit{ is an index permutation}
\end{array}$$

Proof. The proof works just like the proof for Lemma 6 that will be presented in Section 5, except that σ must be a permutation, and the null propagation rule is not needed. □

Example 10. *The simple IND* φ

$$\text{ORDER}[c\_id, d\_id, w\_id] \subseteq_s \text{CUSTOMER}[c\_id, d\_id, w\_id]$$

implies the simple IND

$$\text{ORDER}[w\_id, d\_id, c\_id] \subseteq_s \text{CUSTOMER}[w\_id, d\_id, c\_id]$$

which results from the index permutation $\sigma = \{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 1\}$*. We cannot project attributes away however – the simple IND*

$$\text{ORDER}[w\_id, d\_id] \subseteq_s \text{CUSTOMER}[w\_id, d\_id]$$

resulting from the index projection $\sigma' = \{1 \mapsto 1, 2 \mapsto 2\}$ *is not implied by* φ*, which we already confirmed with the example in the introduction.*

4.3. Non-axiomatizability

Next we consider the real-world case in which simple INDs coexist with not null constraints. Surprisingly, while simple INDs studied in isolation possess an elegant axiomatization (Theorem 3), it turns out that no finite axiomatization exists in the presence of not null constraints. While that this does not preclude the existence of non-finite axiomatizations, we have not been able to identify an elegant one that would aid in human understanding.

Definition 5 (k-ary axiomatization)**.**
We call an inference rule k-ary *for some integer* k *(1-ary, 2-ary, etc.), if and only if it has at most* k *premises. A set of inference rules, or axiom system, is* k*-ary if and only if all its rules are* k*-ary.*

For example, the axiom systems of Theorems 2 and 3 are both 2-ary (and also 3-ary, 4-ary, ...). More generally, all *finite* sets of rules are k-ary for some value k, assuming our rules each take a fixed number of premises.

To establish the following non-axiomatizability result, we will construct, for every k, an example schema where a simple IND φ_0 is implied by $k+1$ INDs, but not by any set containing at most k implied simple INDs other than φ_0. In this we exploit that not null constraints for attributes on the left hand side of an IND force not null restrictions for the matching tuple on the right hand side. By chaining such INDs we obtain INDs where such not null restrictions hold, and are vital for implication of φ_0, but cannot be expressed using simple INDs which might otherwise serve as intermediate steps in a derivation.

Theorem 4. *For the implication of simple inclusion dependencies with not null constraints, no* k*-ary axiomatization exists, for any* k*. In particular, no finite axiomatization exists. This holds even if all relation schemata are restricted to contain at most four attributes.*

Proof. Consider the database schema $\mathcal{R}$ with relation schemata $R_0 = \ldots = R_{k+1} = ABCD$, with $\text{NN}(R_0) = BC$, $\text{NN}(R_{i>0}) = B$. Let further $\Sigma = \{R_i[ABC] \subseteq_s R_{i+1}[ACD] \mid i = 0 \ldots k\}$. It is easy to verify that $\Sigma \vDash \varphi_0 = R_0[A] \subseteq_s R_{k+1}[A]$. We will show that φ_0 cannot be derived using any k-ary set of (sound) derivation rules. To do so, we first need to establish which intermediate INDs could be derived. In the following we only consider non-trivial INDs with ordered LHS attributes.

Claim (*): Let $\varphi = R_n[X] \subseteq_s R_m[Y]$ be an IND implied by Σ, with $n > 0$. Then either $\varphi \in \Sigma$ or we have $m = n + 1$, $X = AC$ and $Y = AD$.

To show this, consider an instance over $\mathcal{R}$ with

$$r_i = \emptyset \text{ for } i < n$$
$$r_n = \left\{ \begin{array}{c} (0, 2, 1, *), \\ (\bot, *, *, *), \\ (*, *, \bot, *) \end{array} \right\}$$
$$r_{n+j} = \{(0, j+2, j+1, j)\} \text{ for } j > 0$$

where each * denotes a unique value. One may verify that Σ holds for this instance. If $D \in X$ then φ is violated by the first tuple in r_n. If $AC \nsubseteq X$ then φ is violated by the second or third tuple in r_n. This leaves $X \in \{AC, ABC\}$. As value 1 only occurs in relations r_n and r_{n+1}, and φ is non-trivial, $m = n+1$. Thus either $\varphi = R_n[AC] \subseteq_s R_{n+1}[AD]$ or $\varphi = R_n[ABC] \subseteq_s R_{n+1}[ACD] \in \Sigma$. This shows Claim (*).

Claim (**): Let $\varphi = R_0[X] \subseteq_s R_m[Y]$ be an IND implied by Σ. Then

$$\varphi \in \left\{ \begin{array}{c} R_0[ABC] \subseteq_s R_1[ACD], \\ R_0[AB] \subseteq_s R_1[AC], \\ R_0[AC] \subseteq_s R_1[AD], \\ R_0[AB] \subseteq_s R_2[AD], \\ R_0[A] \subseteq_s R_m[A] \end{array} \right\}$$

To show this, consider an instance over $\mathcal{R}$ with

$$r_0 = \left\{ \begin{array}{c} (0, 2, 1, *), \\ (\bot, *, *, *) \end{array} \right\}$$
$$r_j = \{(0, j+2, j+1, j)\} \text{ for } j > 0$$

For φ to hold we must have $D \notin X$ (first tuple) and $A \in X$ (second tuple). For $m > 2$, neither 2 nor 1 occurs in r_m, so we must have $X = Y = A$. For $m = 2$ only 2 occurs in r_2, so we must have $X = Y = A$ or $X = AB, Y = AD$. For $m = 1$ the options $X \in \{A, AB, AC, ABC\}$ remain. These are exactly the options listed in Claim (**).

We can summarize claims (*) and (**) as follows: The set

$$\begin{aligned} \Sigma' = \Sigma &\cup \{R_n[AC] \subseteq_s R_{n+1}[AD] \text{ for } n = 0 \ldots k\} \\ &\cup \{R_0[A] \subseteq_s R_{n+1}[A] \text{ for } n = 0 \ldots k\} \\ &\cup \{R_0[AB] \subseteq_s R_1[AC], R_0[AB] \subseteq_s R_2[AD]\} \end{aligned}$$

contains all (non-trivial) INDs implied by Σ over $\mathcal{R}$.

Finally, let $\Sigma'' \subset \Sigma'$ be any set of at most k INDs implied by Σ such that $\varphi_0 \notin \Sigma''$. Then for some $m \in 1, \ldots, k+1$, Σ'' contains no IND with R_m on its RHS. Consider the instance

$$\begin{aligned} r_i &= \{(0, i+2, i+1, i)\} \text{ for } i < m \\ r_i &= \{(0, i+2, \perp, i)\} \text{ for } m \leq i \leq k \\ r_{k+1} &= \emptyset \end{aligned}$$

It is easy to see that all INDs in Σ' which might lie in Σ'' hold for this instance, while φ_0 is violated. This shows that φ_0 cannot be derived using a rule with k or less premises. Hence no k-ary axiomatization exists. □

Example 11. *We illustrate the proof of Theorem 4 for the case $k = 2$. This shows, in particular, that the 2-ary axiomatization for simple INDs from Theorem 3 is not an axiomatization for simple INDs and not null constraints.*

Let $R_0 = R_1 = R_2 = R_3 = ABCD$ with $\text{NN}(R_0) = BC$ and $\text{NN}(R_1) = \text{NN}(R_2) = \text{NN}(R_3) = B$. The set Σ consists of

- $R_0[ABC] \subseteq_s R_1[ACD]$,
- $R_1[ABC] \subseteq_s R_2[ACD]$, *and*
- $R_2[ABC] \subseteq_s R_3[ACD]$.

The following set Σ' contains all non-trivial INDs implied by Σ, up to any permutations applied to both the LHSs and RHSs:

- $R_0[ABC] \subseteq_s R_1[ACD]$, $R_1[ABC] \subseteq_s R_2[ACD]$, $R_2[ABC] \subseteq_s R_3[ACD]$,
- $R_0[AC] \subseteq_s R_1[AD]$, $R_1[AC] \subseteq_s R_2[AD]$, $R_2[AC] \subseteq_s R_3[AD]$,
- $R_0[AB] \subseteq_s R_1[AC]$, $R_0[AB] \subseteq_s R_2[AD]$,
- $R_0[A] \subseteq_s R_1[A]$, $R_0[A] \subseteq_s R_2[A]$, *and* $R_0[A] \subseteq_s R_3[A]$.

Observe in particular that $R_1[A] \subseteq_s R_2[A], R_1[A] \subseteq_s R_3[A]$ and $R_2[A] \subseteq_s R_3[A]$ are not *implied by Σ, as witnessed by the database instance below:*

- $r_0 = \emptyset$,
- $r_1 = \{(0, 0, \perp, 0)\}$,
- $r_2 = \{(1, 1, \perp, 1)\}$, *and*
- $r_3 = \emptyset$.

Let now $\varphi_0 = R_0[A] \subseteq_s R_3[A] \in \Sigma'$. One may that verify φ_0 is not implied by any subset Σ'' of $\Sigma' \setminus \{\varphi_0\}$ containing at most two INDs. As an example, pick the subset $\Sigma'' = \{R_0[AB] \subseteq_s R_2[AD], R_2[AC] \subseteq_s R_3[AD]\}$. Then the database instance below satisfies Σ'' but violates φ_0:

- $r_0 = \{(0, 2, 1, 0)\}$,
- $r_1 = \{(0, 3, \perp, 1)\}$,
- $r_2 = \{(0, 4, \perp, 2)\}$, *and*

- $r_3 = \emptyset$.

Any other choice of Σ'' will show similarly that Σ'' does not imply φ_0. Consequently, φ_0 cannot be derived from Σ by any sound 2-ary rule system.

We have seen that in the presence of not null constraints, partial INDs enjoy a finite axiomatization, while simple INDs do not. As a final case, we consider partial and simple INDs together.

Corollary 1. *For the implication of partial and simple inclusion dependencies with not null constraints, no k-ary axiomatization exists. This holds even if all table schemas are restricted to contain at most four attributes.*

Proof. Assume that such an axiomatization $\mathcal{A}$ existed. Then we can construct an axiomatization $\mathcal{A}'$ by replacing every rule in $\mathcal{A}$ with multiple corresponding rules, in which each partial IND $R[X] \subseteq_p S[Y]$ is replaced by multiple simple INDs, as per Theorem 1, for cardinalities of X up to four. But this would make $\mathcal{A}'$ a k-ary (for some finite k) axiomatization for simple INDs over relation schemata of size four or less, contradicting Theorem 4. □

As simple INDs and not null constraints are the current reality of database practice [31, 48], database research must answer the important question whether there is some suitable extension of simple INDs which *can* be axiomized in the presence of not null constraints, and thus reasoned about more elegantly. Ideally such an extension would also extend partial INDs, thereby unifying the two notions of IND currently proposed by the SQL standard.

In the next section we propose such an extension, and show that it can be reasoned about elegantly and efficiently.

5. Not Null Inclusion Dependencies

In this section we propose *not null inclusion dependencies* (NNINDs) as a new class of inclusion dependencies for SQL-like databases, where attributes may be declared not null. The definition of NNINDs is a natural consequence of the non-axiomatizability result for simple INDs under not null constraints. Both simple and partial INDs are special cases of NNINDs. The name NNIND hails from not null restrictions we impose on participating tuples (e.g. $R[X] \subseteq_s S[Y]$ requires values in X to be not null for the IND to apply).

We establish a finite axiomatization for NNINDs under not null constraints, and the PSPACE-completeness for their implication problem. Therefore, we are able to provide an optimal response to the challenges brought forward by the results of the last section: NNINDs can address the recommendations of the industry standard SQL, and retain the computational properties of its idealized special case of INDs over null-free relations.

The central reason why simple inclusion dependencies cannot be axiomatized is that information about attributes being not null cannot be propagated. Consider e.g. Example 11. Here the IND $R_0[AB] \subseteq_s R_2[AD]$ cannot express the additional requirement, implied by Σ, that the matching tuple over R_2 must not be $\perp$ on attribute C. If it could, then together with $R_2[AC] \subseteq_s R_3[AD]$ it would imply $R_0[A] \subseteq_s R_3[A]$, paving the way for a 2-ary axiomatization.

In the following we will define NNINDs, show how they express partial and simple INDs as special cases, and how they facilitate the propagation of not null restrictions. Note that NNINDs are neither a substitute for not null constraints, nor do they aid in their implication, as discussed in Section 4.1.

Definition 6 (not null inclusion dependency)**.**
Let R be a relation schema, r a relation over R and $U \subseteq R$. We denote by r^U the set of all U-total tuples in r:

$$r^U = \{t \in r \mid \forall A \in U.\, t[A] \neq \perp\}$$

A not null inclusion dependency *(NNIND) is an expression of the form*

$$R^U[X] \subseteq S^V[Y]$$

where R, S are schemata, U, V sets (not necessarily of equal cardinality), and X, Y sequences of equal length with $U, X \subseteq R$ and $V, Y \subseteq S$. We say that such an NNIND holds *on relations r, s over R, S if and only if*

$$r^U[X] \sqsubseteq s^V[Y] \, ,$$

that is, for each projection of a U-total tuple in r on X there is some more informative projection of some V-total tuple in s on Y.

Note that we do not consider repeated attributes in X or Y, in consistency with partial and simple INDs. NNINDs can express both simple and partial INDs as special cases.

Proposition 1. *Partial and simple INDs can be expressed as NNINDs as follows:*

$$R[X] \subseteq_p S[Y] \equiv R^{\emptyset}[X] \subseteq S^{\emptyset}[Y]$$
$$R[X] \subseteq_s S[Y] \equiv R^{X}[X] \subseteq S^{\emptyset}[Y]$$

We briefly illustrate the definition on the following example.

Example 12. *Consider again the database instance presented in Example 2:*

BOOKING

Visitor_id	*Tour_id*	*Site_code*	*Date*
1006	*BRF*	$\perp$	*Sep 19th*
1001	*BRT*	*OR*	*Nov 21st*
1008	$\perp$	*BB*	*Sep 5th*
1012	$\perp$	*MV*	*Nov 2nd*
1011	*RF*	$\perp$	*Oct 5th*

TOUR

Tour_id	*Site_code*	*Site_name*
GCG	*OR*	*O'Reilly's*
BRT	*OR*	*O'Reilly's*
BRT	*MV*	*Movie World*
RF	*BB*	*Binna Burra*
RF	*OR*	*O'Reilly's*

The partial IND

$$\text{Booking}[\mathit{tour\_id}, \mathit{site\_code}] \subseteq_p \text{Tour}[\mathit{tour\_id}, \mathit{site\_code}]$$

can be expressed by the NNIND

$$\text{Booking}^{\emptyset}[\mathit{tour\_id}, \mathit{site\_code}] \subseteq \text{Tour}^{\emptyset}[\mathit{tour\_id}, \mathit{site\_code}]$$

but is violated by the instance, as the first row in Booking *has no matching tuple in* Tour. *The simple IND*

$$\text{Booking}[\mathit{tour\_id}, \mathit{site\_code}] \subseteq_s \text{Tour}[\mathit{tour\_id}, \mathit{site\_code}]$$

can be expressed by the NNIND

$$\text{Booking}^{\mathit{tour\_id},\mathit{site\_code}}[\mathit{tour\_id}, \mathit{site\_code}] \subseteq \text{Tour}^{\emptyset}[\mathit{tour\_id}, \mathit{site\_code}]$$

and holds on the instance. It can further be strengthened to the NNIND

$$\text{Booking}^{\mathit{site\_code}}[\mathit{tour\_id}, \mathit{site\_code}] \subseteq \text{Tour}^{\emptyset}[\mathit{tour\_id}, \mathit{site\_code}]$$

which is still satisfied by the instance.

As we have seen, NNINDs generalize partial and simple INDs in a unified framework. In the example above, the sets U was a subset of X, and the set V was empty. NNINDs of this form can actually be expressed by a set of simple INDs, generalizing Theorem 1:

Corollary 2. *Every NNIND of the form $R^U[UX] \subseteq S^{\emptyset}[Y]$ is semantically equivalent to a set of simple INDs (for both finite and unrestricted implication):*

$$R^U[UX] \subseteq S^{\emptyset}[Y] \;\equiv\; \{\, R[UX'] \subseteq_s S[Y'] \;\mid\; X' \subseteq X,\; Y' = (UX \mapsto Y)(UX') \,\}$$

However, while such NNIND are sufficient to generalize both partial and simple INDs, they are not sufficient to allow axiomatization. As we observed in the proof of Theorem 4, we need (at the very least) the ability to express that attributes not in Y must not be null in the matching tuple.

Our central result for this section will be a (finite) axiomatization of NNINDs. This axiomatization is given in Table 1. The first three rules simply extend the axioms for simple and partial INDs. Note that rule (P) allows projection, even when dealing with NNINDs repesenting simple INDs, as the set U allows us to maintain the not null restrictions for attributes dropped in the projection.

Rule (N) enables us to propagate not null restrictions, such as those observed in the proof of Theorem 4. Note that $(X \mapsto Y)(U)$ is the set of attributes matching $U \cap X$ in S, as per Definition 3, so for any U-total tuple t over R, any matching tuple s over S with $r[X] \sqsubseteq s[Y]$ must not be null on $(X \mapsto Y)(U)$.

Rule (D) captures the relationship between simple and partial INDs, as per Theorem 1. As we will see in Lemma 6, we can avoid rule (D) if we restrict ourselves to a subclass of NNINDs which generalize simple but not partial INDs.

We will use the symbol $\vdash$ to denote derivability using the axioms in Table 1.

$$\frac{}{R^U[X] \subseteq R^V[X]}\ V \subseteq U \cup \mathrm{NN}(R) \qquad \frac{R^U[X] \subseteq S^V[Y] \quad S^V[Y] \subseteq T^W[Z]}{R^U[X] \subseteq T^W[Z]}$$

Reflexivity (R) — *Transitivity (T)*

$$\frac{R^U[X] \subseteq S^V[Y]}{R^U[\sigma X] \subseteq S^V[\sigma Y]}\ \sigma \text{ is an IPP} \qquad \frac{R^U[X] \subseteq S^V[Y]}{R^U[X] \subseteq S^W[Y]}\ W = V \cup (X \mapsto Y)(U)$$

Projection & Permutation (P) — *Not Null Propagation (N)*

$$\frac{R^{UA}[AX] \subseteq S^V[BY] \quad R^U[X] \subseteq S^V[Y]}{R^U[AX] \subseteq S^V[BY]}$$

Tertium Non Datur (D)

Table 1: Axiomatization of not null Inclusion Dependencies

Theorem 5. *The set of axioms in Table 1 is sound and complete for not null inclusion dependencies.*

To establish some results needed to prove Theorem 5, we introduce the auxiliary notion of quasi-simple NNINDs. Quasi-simple NNINDs can be axiomatized without use of the Tertium Non Datur Rule, and completeness of this axiomatization is easier to establish.

5.1. Quasi-simple Not Null Inclusion Dependencies

Inclusion dependencies with simple semantics only apply to tuples not containing any null values in LHS attributes. We extend this notion to NNINDs, and shall call such dependencies quasi-simple.

Definition 7 (quasi-simple NNIND)**.**
We call NNINDs of the form $R^{UX}[X] \subseteq S^V[Y]$ quasi-simple*.*

Example 13. *The NNIND*

$$\textsc{Booking}^{tour\_id, site\_code}[tour\_id, site\_code] \subseteq \textsc{Tour}^{\emptyset}[tour\_id, site\_code]$$

is quasi-simple, while the NNIND

$$\textsc{Booking}^{site\_code}[tour\_id, site\_code] \subseteq \textsc{Tour}^{\emptyset}[tour\_id, site\_code]$$

is not quasi-simple.

One may note that quasi-simple NNINDs include simple INDs. The relationship between partial and simple INDs extends to NNINDs and quasi-simple INDs (with the difference that NNINDs include quasi-simple NNINDs, whereas partial INDs do not include simple ones):

Theorem 6. *Every NNIND is semantically equivalent to a set of quasi-simple NNINDs. Specifically:*

$$R^U[X] \subseteq S^V[Y] \equiv \{\, R^{UX'}[X'] \subseteq S^V[Y'] \mid X' \subseteq X,\ Y' = (X \mapsto Y)(X') \}$$

Proof. It is easy to see that $R^U[X] \subseteq S^V[Y]$ implies every $R^{UX'}[X'] \subseteq S^V[Y']$ in the set.

To show the reverse, let r, s be instances of R, S so that every $R^{UX'}[X'] \subseteq S^V[Y']$ holds on r, s, and $t \in r$ be U-total. From $R^{UX'}[X'] \subseteq S^V[Y']$ with $X' = \mathrm{NN}(t) \cap X$, it follows that there exists a V-total $t' \in s$ with $t[X'] = t'[Y']$. But this means $t[X] \sqsubseteq t'[Y]$, and since such a t' exists for all U-total $t \in r$, $R[X] \subseteq_p S[Y]$ is satisfied. □

Example 14. *The NNIND*

$$\textsc{Booking}^{site\_code}[tour\_id, site\_code] \subseteq \textsc{Tour}^{\emptyset}[tour\_id, site\_code]$$

is not quasi-simple, but semantically equivalent to the following set of quasi-simple NNINDs:

- $\textsc{Booking}^{tour\_id, site\_code}[tour\_id, site\_code] \subseteq \textsc{Tour}^{\emptyset}[tour\_id, site\_code]$,
- $\textsc{Booking}^{tour\_id, site\_code}[tour\_id] \subseteq \textsc{Tour}^{\emptyset}[tour\_id]$,
- $\textsc{Booking}^{site\_code}[site\_code] \subseteq \textsc{Tour}^{\emptyset}[site\_code]$, *and*
- $\textsc{Booking}^{site\_code}[\,] \subseteq \textsc{Tour}^{\emptyset}[\,]$.

Note that the 2nd and 4th NNIND in this list are implied by the 1st and 3rd, respectively, and thus redundant. This reduced representation could also be obtained via Corollary 2, by replacing simple INDs with quasi-simple NNINDs.

5.2. Chase for NNINDs

The Chase is often used as an efficient algorithm for deciding instances of implication problems. Apart from being useful in its own right, we will use the chase to show completeness of our axiomatization, and later to establish the fixed parameter-tractability of NNINDs in their arity.

As implication of an NNIND reduces to implication of a set of quasi-simple NNINDs (of bounded cardinality for NNINDs of bounded arity) by Theorem 6, we will only consider a chase for quasi-simple NNINDs.

Algorithm 1 (Chase for quasi-simple NNINDs).
Input: *A database schema $(\mathcal{R}, \Sigma)$ with $\mathcal{R} = (R_1, \dots, R_k)$ and a finite database $\boldsymbol{r} = (r_1, \dots, r_k)$ over $\mathcal{R}$.*
Output: *A modified database chase$(\boldsymbol{r}, \Sigma)$.*
Method: *Apply the following rule as long as possible:*

`NNIND`: *If Σ contains a NNIND $R^U[X] \subseteq S^V[Y]$ and for some U-total tuple t_R over R there does not exist a V-total tuple t_S over S with $t_R[X] \sqsubseteq t_S[Y]$, then add the following tuple t_S over S:*

$$t_S[B_i \in Y] = \begin{cases} 0 & \text{if } t_R[A_i] = \bot \text{ and } B_i \in V \cup \mathrm{NN}(S) \\ t_R[A_i] & \text{otherwise} \end{cases}$$

$$t_S[B \notin Y] = \begin{cases} 0 & \text{if } B \in V \cup \mathrm{NN}(S) \\ \bot & \text{otherwise} \end{cases}$$

Here we use the notation $A_i = (Y \mapsto X)(B_i)$ for readability, and say that t_S was added as a result of $R^U[X] \subseteq S^V[Y]$ applied to t_R.

Lemma 3. *Algorithm 1 terminates, and Σ holds on chase($\boldsymbol{r}, \Sigma$).*

Proof. As the initial database r is finite, the set $\mathfrak{A}_0$ of attribute values occurring in r is finite. Application of rule `NNIND` introduces at most two new attribute values, 0 and $\perp$, so the set $\mathfrak{A} = \mathfrak{A}_0 \cup \{0, \perp\}$ of attribute values occurring in any intermediate database is finite as well.

It follows that the set of databases over $\mathcal{R}$ containing only values from alphabet $\mathfrak{A}$ is finite. As each application of rule `NNIND` strictly increases the number of tuples in the database, only a finite number of rule applications is possible, so Algorithm 1 terminates.

The resulting database satisfies Σ, as otherwise rule `NNIND` could be applied again, contrary to our termination criterion. □

While the chase can be applied to arbitrary input databases, we will want to pick our input database in such a way that the result allows us to reason about the implication of a given quasi-simple NNIND φ.

Definition 8 (Chase for (φ, Σ))**.**
Let Σ be a set of NNINDs over $\mathcal{R}$, and $\varphi = R^{UX}[X] \subseteq S^V[Y]$ be a quasi-simple NNIND over $\mathcal{R}$. We write chase(φ, Σ) *to denote chase($\boldsymbol{r}, \Sigma$), where the initial database $\boldsymbol{r}$ consist of a single tuple t_0 over R with*

$$t_0[X] = (1, \dots, m) \quad \textit{where } m = |X|$$

$$t_0[A \notin X] = \begin{cases} 0 & \textit{for } A \in U \cup \mathrm{NN}(R) \\ \perp & \textit{otherwise} \end{cases}$$

Lemma 4. *Let Σ, φ, t_0 be as in Definition 8. If chase(φ, Σ) contains a tuple t over R_j with $t_0[X] = t[E]$ for some sequence E over R_j, then*

$$R^{UX}[X] \subseteq {R_j}^{\mathrm{NN}(t)}[E]$$

can be derived from Σ using the first four axioms of Table 1.

Proof. We proceed by induction on the sequence of tuple additions.

For $t = t_0$ derivability follows by the reflexivity axiom. So let t be added as a result of $\phi = {R_i}^{U^*}[X^*] \subseteq {R_j}^{V^*}[Y^*]$ applied to t'. By construction this means

$$\mathrm{NN}(t) = V^* \cup \mathrm{NN}(R_j) \cup (X^* \mapsto Y^*)(\mathrm{NN}(t'))$$

From $t_0[X] = t[E]$ it follows that $E \subseteq Y^*$, as all values in t outside Y^* are 0 or $\perp$. Hence there exists an IPP σ such that $E = \sigma Y^*$. We thus obtain $t_0[X] = t[E] = t[\sigma Y^*] = t'[\sigma X^*]$. Thus by induction hypothesis, we have

$$\Sigma \vdash R^{UX}[X] \subseteq {R_i}^{\mathrm{NN}(t')}[\sigma X^*]$$

Recall that $\vdash$ denotes derivability using the axioms of Table 1, and in this instance using only the first four axioms.

Furthermore, as rule `NNIND` was applicable, we must have $U^* \subseteq \text{NN}(t')$. We can thus derive

$$\dfrac{\dfrac{\vdots}{R^{UX}[X] \subseteq {R_i}^{\text{NN}(t')}[\sigma X^*]} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{{R_i}^{U^*}[X^*] \subseteq {R_j}^{V^*}[Y^*]}{{R_i}^{\text{NN}(t')}[X^*] \subseteq {R_j}^{V^*}[Y^*]}\,(R{+}T)}{{R_i}^{\text{NN}(t')}[X^*] \subseteq {R_j}^{V^* \cup \text{NN}(R_j)}[Y^*]}\,(R{+}T)}{{R_i}^{\text{NN}(t')}[X^*] \subseteq {R_j}^{\text{NN}(t)}[Y^*]}\,(N)}{{R_i}^{\text{NN}(t')}[\sigma X^*] \subseteq {R_j}^{\text{NN}(t)}[\sigma Y^*]}\,(P)}{R^{UX}[X] \subseteq {R_j}^{\text{NN}(t)}[\sigma Y^*]}\,(T)$$

As $E = \sigma Y^*$, this completes the proof. □

As shown next, the chase can be used to decide implication of quasi-simple NNINDs (by a set of arbitrary NNINDs) algorithmically.

Lemma 5. *Let Σ, φ, t_0 be as in Definition 8. Then $\Sigma \models \varphi$ if and only if chase(φ, Σ) contains a V-total tuple t_S over S with $t_0[X] = t_S[Y]$.*

Proof. If such a tuple t_S exists, then by Lemma 4 we can derive φ from Σ using the first four axioms of Table 1. Since these are sound (as one may easily verify), this shows $\Sigma \models \varphi$.

If no such tuple t_S exists, then chase(φ, Σ) violates φ. By Lemma 3 Σ holds on chase(φ, Σ), showing that Σ does not imply φ. □

As mentioned earlier, we can also apply Algorithm 1 to decide the implication problem for general NNINDs, by employing the relationship between general NNINDs and quasi-simple NNINDs established in Theorem 6.

We illustrate the chase on one of our running examples.

Example 15. *In Example 1 we claimed that the simple INDs*

$$\varphi : \textsc{Talk}[\textit{CName}, \textit{PName}] \subseteq_s \textsc{Chair}[\textit{CName}, \textit{PName}]$$

is implied by the partial and simple INDs

$$\begin{aligned}
\varphi_1 &: \textsc{Chair}[\textit{CName}] \subseteq_p \textsc{Conference}[\textit{CName}]\\
\varphi_2 &: \textsc{Chair}[\textit{PName}] \subseteq_p \textsc{Participant}[\textit{PName}]\\
\varphi_3 &: \textsc{Booking}[\textit{CName}, \textit{PName}] \subseteq_s \textsc{Chair}[\textit{CName}, \textit{PName}]\\
\varphi_4 &: \textsc{Talk}[\textit{CName}, \textit{PName}] \subseteq_p \textsc{Booking}[\textit{CName}, \textit{PName}]
\end{aligned}$$

We shall test this claim using Lemma 5, representing simple/partial INDs as NNINDs, and abbreviating attributes by their first character:

$$\varphi : \textsc{Talk}^{CP}[CP] \subseteq \textsc{Chair}^{\emptyset}[CP]$$
$$\varphi_1 : \textsc{Chair}^{\emptyset}[C] \subseteq \textsc{Conference}^{\emptyset}[C]$$
$$\varphi_2 : \textsc{Chair}^{\emptyset}[P] \subseteq \textsc{Participant}^{\emptyset}[P]$$
$$\varphi_3 : \textsc{Booking}^{CP}[CP] \subseteq \textsc{Chair}^{\emptyset}[CP]$$
$$\varphi_4 : \textsc{Talk}^{\emptyset}[CP] \subseteq \textsc{Booking}^{\emptyset}[CP]$$

To compute chase$(\varphi, \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\})$ *we start with the initial database containing only a single tuple* $t_0 = (1, 2, 0)$ *over* Talk. *Applying Algorithm 1 to this database results in*

Talk		
C	*P*	*T[NN]*
1	*2*	*0*

Booking			
C	*P*	*D[NN]*	*R[NN]*
1	*2*	*0*	*0*

Chair	
C[NN]	*P[NN]*
1	*2*

Conference
C[NN]
1

Participant
P[NN]
2

As Chair *contains tuple* $(1, 2)$, *our claim holds by Lemma 5.*

We further claimed in Example 1 that (using NNIND representation)

$$\varphi' : \textsc{Talk}^{C}[C] \subseteq \textsc{Conference}^{\emptyset}[C]$$

is not implied by $\{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ *under the not null constraints specified. Again we can test this using Lemma 5, starting with the initial database containing only a single tuple* $t_0 = (1, \perp, 0)$ *over* Talk. *Applying Algorithm 1 to this database results in*

Talk		
C	*P*	*T[NN]*
1	$\perp$	*0*

Booking			
C	*P*	*D[NN]*	*R[NN]*
1	$\perp$	*0*	*0*

Chair	
C[NN]	*P[NN]*

Conference
C[NN]

Participant
P[NN]

Since Conference *does not contain the tuple* $t = (1)$, *it follows by Lemma 5 that* φ' *is not implied, as claimed.*

For partial INDs and INDs over null-free relations, specialized *chase* algorithms can be found in [1, 38].

5.3. Completeness Proofs

We now have the necessary foundations to show that our axiomatization is complete. We first show completeness for quasi-simple NNINDs, and then extend this to general NNINDs.

Lemma 6. *The first four rules of Table 1 are sound, and complete for the derivation of quasi-simple NNINDs.*

Proof. It is easy to verify that the axioms are sounds.

Our proof that any implied quasi-simple NNIND can be derived proceeds along the lines of that for standard INDs [15, Theorem 3.1].

Let Σ, φ, t_0 be as in Definition 8, with $\Sigma \vDash \varphi = R^U[X] \subseteq S^V[Y]$. Then by Lemma 5 chase$(\varphi, \Sigma)$ contains a V-total tuple t_S over S with $t_0[X] = t_S[Y]$. From this it follows by Lemma 4 that $\varphi' = R^{UX}[X] \subseteq S^{\mathrm{NN}(t_S)}[Y]$ can be derived from Σ using the first four axioms. As $V \subseteq \mathrm{NN}(t_S)$ we can derive φ from φ' with another application of reflexivity and transitivity axioms. □

We illustrate the use of our inference rules by proving again the first claim of Examples 1. Note that non-implication of (NN)INDs, in particular the second claim of Example 1, cannot be shown this way.

Example 16. *In Example 1 we claimed that the simple IND*

$$\varphi : \textsc{Talk}[CName, PName] \subseteq_s \textsc{Chair}[CName, PName]$$

is implied by the given set Σ and the not null constraints. We now provide an inference to substantiate the claim, using the first four rules of Table 1. The attributes involved are abbreviated by their first letters.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\textsc{Talk}^{CP}[CP] \subseteq \textsc{Talk}^{\emptyset}[CP]}\,(R) \qquad \textsc{Talk}^{\emptyset}[CP] \subseteq \textsc{Booking}^{\emptyset}[CP]}{\textsc{Talk}^{CP}[CP] \subseteq \textsc{Booking}^{\emptyset}[CP]}\,(T)}{\textsc{Talk}^{CP}[CP] \subseteq \textsc{Booking}^{CP}[CP]}\,(N) \qquad \textsc{Booking}^{CP}[CP] \subseteq \textsc{Chair}^{\emptyset}[CP]}{\textsc{Talk}^{CP}[CP] \subseteq \textsc{Chair}^{\emptyset}[CP]}\,(T)$$

The final NNIND in this example can be rewritten as the simple IND φ.

We are now ready to show Theorem 5.

Proof of Theorem 5. Soundness of Tertium Non Datur (D) may not be obvious, so we provide a brief proof. Consider any U-total tuple t_R over R.

If $t[A] \neq \bot$ then existence of a V-total t_S over S with $t_R[AX] \sqsubseteq t_S[BY]$ follows from $R^{UA}[AX] \subseteq S^V[BY]$.

If $t_R[A] = \bot$ then by $R^U[X] \subseteq S^V[Y]$ there exists a V-total t_S over S with $t_R[X] \sqsubseteq t_S[Y]$. With $t_R[A] = \bot$ it follows that $t_R[AX] \sqsubseteq t_S[BY]$.

To show completeness, denote again our relation schemata by $R_1, \ldots, R_n$. Let Σ be a set of NNINDs, and

$$\varphi = R^U[X] \subseteq S^V[Y]$$

some NNIND with $\Sigma \vDash \varphi$. We show that for all IPP σ and $W \subseteq \sigma X$ we have

$$\Sigma \vdash \varphi_\sigma^W = R^{U(\sigma X \setminus W)}[\sigma X] \subseteq S^V[\sigma Y] \tag{1}$$

We proceed by induction on the cardinality of W.

Let $|W| = 0$. Then $\Sigma \vDash \varphi \vDash \varphi_\sigma^\emptyset$, and our claim follows by Lemma 6.

So assume (1) holds for all $|W| \leq n$, and let $|W'| = n+1$ and σ some IPP. Pick $A \in W'$ so that $W' = AW$ with $|W| = n$, and let σ_A be derived from σ by "dropping" A. By (1) we have $\Sigma \vdash \varphi_\sigma^W$ and $\Sigma \vdash \varphi_{\sigma_A}^W$. From

$$\begin{aligned} \varphi_\sigma^W &= R^{U(\sigma X \setminus W')A}[\sigma X] \subseteq S^V[\sigma Y] \quad \text{and} \\ \varphi_{\sigma_A}^W &= R^{U(\sigma X \setminus W')}[\sigma_A X] \subseteq S^V[\sigma_A Y] \end{aligned}$$

we can derive, using Tertium Non Datur (plus Permutation to make A the first attribute):

$$\varphi_\sigma^{W'} = R^{U(\sigma X \setminus W')}[\sigma X] \subseteq S^V[\sigma Y] \ .$$

This shows (1) which includes $\Sigma \vdash \varphi$ as a special case. $\square$

The following example illustrates the Tertium Non Datur rule.

Example 17. *The database instance from Example 2 satisfies the NNIND*

$$\text{Booking}^{site\_code}[tour\_id, site\_code] \subseteq \text{Tour}^\emptyset[tour\_id, site\_code]$$

but violates the NNIND (partial IND)

$$\text{Booking}^\emptyset[tour\_id, site\_code] \subseteq \text{Tour}^\emptyset[tour\_id, site\_code] \ .$$

Soundness of the Tertium Non Datur rule implies, in particular, that

$$\text{Booking}^\emptyset[tour\_id] \subseteq \text{Tour}^\emptyset[tour\_id]$$

must also be violated by the database instance. Indeed, the tour_id-value BRF occurs in the Booking*-table, but not in the* Tour*-table.*

By Lemma 6, the Tertium Non Datur rule is not needed to derive simple INDs. Nor is it required to derive partial INDs from a set of partial INDs, as per Theorem 2. While it is necessary for completeness using our definition of NNINDs, one may thus wonder whether it could be avoided by using a more conservative generalisation of partial and simple INDs. Theorem 1 suggests that this is not possible, and the following example confirms this (it is of course possible to replace it by some other rule, though we suspect that any replacement rule would be very similar and not simpler).

Example 18. *Let $R = AB$, $S = CD$ be relation schemata without not null constraints, and*

$$\Sigma = \{\ R^A[A] \subseteq S[C],\ R[B] \subseteq S[D]\ \}$$

a set containing only simple and partial INDs. Then Σ implies the partial IND $R[A] \subseteq S[C]$, as shown below:

$$\dfrac{R^A[A] \subseteq S[C] \qquad \dfrac{R[B] \subseteq S[D]}{R[\,] \subseteq S[\,]}\,(P)}{R[A] \subseteq S[C]}\,(D)$$

One may verify that $R[A] \subseteq S[C]$ cannot be derived without use of rule (D). Note that $R[\,] \subseteq S[\,]$ is non-trivial, and expresses that S is non-empty if R is non-empty.

5.4. Hardness of Implication

It has been shown in [15, Theorem 3.3] that implication of INDs for null-free relations is PSPACE-complete. As NNINDs extend INDs over null-free relations, implication between NNINDs is at least PSPACE-hard. We will show that it lies in PSPACE next.

The following lemma matches [15, Theorem 3.3], and shows PSPACE-computability for quasi-simple NNINDs.

Lemma 7. *Let Σ be a set of (general) NNINDs and $\varphi = R^{UX}[X] \subseteq S^V[Y]$ a quasi-simple NNIND. Then $\Sigma \vDash \varphi$ iff there exists a sequence*

$$R^{UX}[X] = S_0^{U_0}[X_0],\ \ldots,\ S_n^{U_n}[X_n] = S^V[Y]$$

where S_i are relation schemata, and $U_i, X_i \subseteq S_i$ such that for each $0 \leq i < n$ the NNIND

$$S_i{}^{U_i}[X_i] \subseteq S_{i+1}{}^{U_{i+1}}[X_{i+1}]$$

is either an instance of the reflexivity axiom (R) or can be derived (non-deterministically in linear space) from a single NNIND in Σ.

Proof. The "if" direction is obvious. For the "only if" direction, consider the derivation trees constructed in the proof of Lemma 4, copied below:

$$\dfrac{\dfrac{\vdots}{R^{UX}[X] \subseteq R_i{}^{\mathrm{NN}(t')}[\sigma X^*]} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{R_i{}^{U^*}[X^*] \subseteq R_j{}^{V^*}[Y^*]}{R_i{}^{\mathrm{NN}(t')}[X^*] \subseteq R_j{}^{V^*}[Y^*]}\,(R{+}T)}{R_i{}^{\mathrm{NN}(t')}[X^*] \subseteq R_j{}^{V^* \cup \mathrm{NN}(R_j)}[Y^*]}\,(R{+}T)}{R_i{}^{\mathrm{NN}(t')}[X^*] \subseteq R_j{}^{\mathrm{NN}(t)}[Y^*]}\,(N)}{R_i{}^{\mathrm{NN}(t')}[\sigma X^*] \subseteq R_j{}^{\mathrm{NN}(t)}[\sigma Y^*]}\,(P)}{R^{UX}[X] \subseteq R_j{}^{\mathrm{NN}(t)}[\sigma Y^*]}\,(T)$$

Our $S_i^{U_i}[X_i]$ for $i > 1$ are the RHSs

$$\ldots,\ R_i^{\mathrm{NN}(t')}[\sigma X^*],\ R_j^{\mathrm{NN}(t)}[\sigma Y^*],\ \ldots$$

of the NNINDs derived, and each "connecting" NNIND

$$R_i{}^{\mathrm{NN}(t')}[\sigma X^*] \subseteq R_j{}^{\mathrm{NN}(t)}[\sigma Y^*]$$

is derived from the respective $R_i{}^{U^*}[X^*] \subseteq R_j{}^{V^*}[Y^*] \in \Sigma$, in linear space. □

Using the relationship between quasi-simple and general NNINDs established in Theorem 6, we can now derive PSPACE-completeness for the latter.

Theorem 7. *The implication problem for not null inclusion dependencies is PSPACE-complete.*

Proof. PSPACE-hardness follows from PSPACE-hardness for INDs over null-free relations [15, Theorem 3.3].

It remains to show that it lies in PSPACE. From Lemma 7 it follows that there exists a non-deterministic algorithm that decides derivability of quasi-simple NNINDs in linear space. By Theorem 6 we can check implication of a non-quasi-simple NNIND by sequentially checking quasi-simple NNINDs, again in linear space (while the number of quasi-simple NNINDs is exponential, they can be enumerated in linear space). This provides us with a linear space non-deterministic algorithm. Existence of a deterministic polynomial space algorithm follows from Savitch's Theorem [53]. □

6. Expressive fragments of NNIND with lower complexity

In the case of null-free relations, the PSPACE-completeness result for deciding the implication of INDs has motivated researchers to identify expressive fragments whose implication problem is less complex to decide. In this section, we pursue the same goal for NNINDs. In particular, we show that the implication problems for i) typed acyclic NNINDs, and ii) typed acyclic simple and partial INDs with not null constraints are each NP-hard to decide, the implication problem for NNINDs is fixed parameter-tractable in their arity, and the implication problem for tree-like NNINDs can be decided in linear time.

6.1. Intractable Subclasses

As an example of an intractable subclass of NNINDs we consider typed NNINDs, which are commonly encountered in database practice. Interestingly, it turns out that the complexity of the implication problem of typed NNINDs differs from that of typed INDs over null-free relations. In [16] it was shown that for typed INDs over null-free relations, implication can be decided in polynomial time. For typed NNINDs the associated implication problem is NP-hard, already in the case where the given set of NNINDs is acyclic.

Definition 9 (typed NNINDs)**.**
We call an NNIND $R^U[X] \subseteq S^V[Y]$ typed *if and only if $X = Y$.*

Definition 10 (acyclic NNINDs)**.**
We call a set Σ of NNINDs acyclic *if and only if the directed graph $\mathcal{G}$ with the relation schemata as nodes and an edge $R \to S$ for every NNIND $R^U[X] \subseteq S^U[Y] \in \Sigma$ is acyclic.*

Example 19. *The INDs of Example 1 are both typed and acyclic. Consider now relation schema* Employee*(EmpName[NN], Manager). The IND*

$$\varphi : \textsc{Employee}[\mathit{Manager}] \subseteq_s \textsc{Employee}[\mathit{EmpName}]$$

is untyped, and expresses that the manager of an employee must also be an employee. The set $\{\varphi\}$ is cyclic, inducing a cycle of length 1.

Theorem 8. *The implication problem for typed acyclic NNINDs is NP-hard.*

Proof. We will reduce the conjunctive normal form satisfiability problem (CNF-SAT) to it.

Let $\mathcal{V}$ be a set of boolean variables and $\mathcal{C} = \{c_1, \ldots, c_m\}$ a set of clauses over $\mathcal{V} = \{v_1, \ldots, v_n\}$. We construct a schema $\mathcal{R} = \{R_0, \ldots, R_{n+1}\}$ and an acylic set $\Sigma = \{\varphi_{-n}, \ldots, \varphi_n\}$ of typed NNINDs over $\mathcal{R}$ as follows.

- $R_i = X = \{A_1, \ldots, A_m\}$ for $i = 0, \ldots, n+1$
- $\varphi_i = {R_{i-1}}^{\emptyset}[X] \subseteq {R_i}^{V_i}[X]$ for $i = 1, \ldots, n$ with $V_i = \{A_j \mid v_i \in c_j\}$
- $\varphi_{-i} = {R_{i-1}}^{\emptyset}[X] \subseteq {R_i}^{V_{-i}}[X]$ for $i = 1, \ldots, n$ with $V_{-i} = \{A_j \mid \neg v_i \in c_j\}$
- $\varphi_0 = {R_n}^{X}[X] \subseteq {R_{n+1}}^{\emptyset}[X]$

We claim that Σ implies $\varphi = {R_0}^{\emptyset}[X] \subseteq {R_{n+1}}^{\emptyset}[X]$ iff $\mathcal{C}$ is satisfiable.

Let $\mathcal{S}$ be any derivation sequence that derives φ from Σ. Such a sequence would necessarily consist of exactly one of φ_i, φ_{-i} for $i = 1, \ldots, n$, followed by φ_0. In order for φ_0 to be applicable, a "not null" condition must be derived for each A_j. For this to happen, at least one φ_i or φ_{-i} must occur in $\mathcal{S}$ with $A_j \in V_i$ or $A_j \in V_{-i}$, respectively.

Consider now the truth assignment $\mathcal{T}_{\mathcal{S}}$:

$$\mathcal{T}_{\mathcal{S}}(v_i) = \begin{cases} true & \text{if } \varphi_i \in \mathcal{S} \\ false & \text{if } \varphi_{-i} \in \mathcal{S} \end{cases}$$

which satisfies a clause $c_j \in \mathcal{C}$ iff $\varphi_i \in \mathcal{S}$ for some $v_i \in c_j$ or $\varphi_{-i} \in \mathcal{S}$ for some $\neg v_i \in c_j$. By definition of V_i and V_{-i} we have $v_i \in c_j$ iff $A_j \in V_i$ and $\neg v_i \in c_j$ iff $A_j \in V_{-i}$.

Thus, if a derivation sequence $\mathcal{S}$ exists showing $\Sigma \vDash \varphi$, then $\mathcal{T}_{\mathcal{S}}$ is a truth assignment satisfying $\mathcal{C}$. Conversely, every truth assignment $\mathcal{T}$ satisfying $\mathcal{C}$ gives rise to a derivation sequence $\mathcal{S}$ with $\mathcal{T} = \mathcal{T}_{\mathcal{S}}$ showing $\Sigma \vDash \varphi$. □

While the proof of Theorem 8 uses NNINDs with non-empty sets V, this is not strictly necessary. We could easily avoid this by replacing each $\varphi_i \neq \varphi_0$ with a pair of partial INDs $R_{i-1}[X] \subseteq_p S_i[X]$ and $S_i[X] \subseteq_p R_i[X]$ by adding an additional schema S_i with not null constraints on V_i, and respectively for φ_{-i}. This gives us the following corollary.

Corollary 3. *The implication problem for typed acyclic simple and partial inclusion dependencies under not null constraints is NP-hard to decide.*

We remark that by Corollary 3 the increased difficulty in deciding implication of typed NNINDs cannot be blamed on our choice in defining NNINDs, but is intrinsic to simple and partial IND in combination with not null constraints.

6.2. Tractable Subclasses

In [15, 22] it was shown that implication for inclusion dependencies of bounded arity over null-free relations can be decided in polynomial time. As we will show next, this result not only extends to NNINDs, but can actually be strengthened to show fixed parameter-tractability (FPT).

Definition 11 (fixed parameter tractability, [25])**.**
Let n denote the size of a problem instance. A problem is fixed-parameter tractable in parameter k if and only if it can be solved (deterministically) in time $P(n) \cdot f(k)$, for some polynomial P and some function f.

Note that being fixed-parameter tractable in k is strictly stronger than being polynomial for every fixed k. E.g. a runtime of $O(n^k)$ is polynomial for every fixed k, but the degree of these polynomials is unbounded.

Definition 12 (arity)**.**
The arity *of a NNIND $R^U[X] \subseteq S^V[Y]$ is the length of X (equivalently Y). When we talk of implication for bounded arity, we mean implication over subclasses of NNINDs induced by a fixed upper bound on their arity.*

One would expect INDs to have small arity in practice, making it a sensible parameter to fix.

Theorem 9. *The implication problem for NNINDs is fixed parameter-tractable in their arity.*

Proof. Let the arity of all NNINDs in $\Sigma \cup \{\varphi\}$ be bounded by m and consider Algorithm 1 for quasi-simple NNINDs (i.e., where φ is quasi-simple). The number of distinct attribute values that may occur in tuples produced during our chase is bounded by $m + 2$.

For each tuple generated, there exists some NNIND $\varphi' = {R_i}^{U^*}[X^*] \subseteq {R_j}^{V^*}[Y^*] \in \Sigma$ that generates it. Consider now the tuples generated by φ'. By construction they only differ on Y^*, and thus their number is bounded by $(m+2)^{|Y^*|} \leq (m+2)^m$. Hence the total number of tuples generated during our chase is bounded by $|\Sigma| \cdot (m+2)^m$. Since the number of chase steps (successful or not) is limited by $|\Sigma|$ times number of tuples occurring in our chase tables, and each chase step can be implemented to run in polynomial time, this means that running time for our chase is bounded by $P(\text{input size}) \cdot (m+2)^m$ for some polynomial P.

Finally, we can decide implication of a non-quasi-simple NNIND of arity $\leq m$ by checking implication of at most 2^m quasi-simple NNINDs by Theorem 6. This gives us an overall running time bounded by

$$P(\text{input size}) \cdot (2m+4)^m$$

which shows fixed parameter-tractability in m. □

Another tractable class of INDs are *tree-like* INDs, meaning that $\mathcal{G}$ in Definition 10 is a tree (or forest) without duplicate edges. Implication for tree-like INDs over null-free relations can be decided in linear time [41], and the proof (omitted) generalizes to NNINDs over relations with not null constraints.

Theorem 10. *For tree-like NNINDs the implication problem can be decided in linear time.* □

Example 20. *The directed graph induced by the INDs of Example 1 is a tree. As seen in Example 15, this guarantees that Algorithm 1, if applied to an initial database containing a single tuple, produces at most one tuple per relation.*

7. On the interaction of NNINDs and functional dependencies

While inclusion dependencies provide a general framework to accommodate referential integrity, functional dependencies (FDs) provide a general framework to accommodate entity integrity. As both integrity rules are fundamental to data management, the interaction of functional and inclusion dependencies is highly interesting.

Already in null-free relational databases, unfortunately, the implication problem for the combined class of functional and inclusion dependencies is undecidable [49, 17]. More precisely, finite and unrestricted implication problems for this combined class are different, and each of them is undecidable. It was further shown that the implication of

1. typed acyclic INDs and FDs together is NP-hard [21],
2. typed INDs is possible in polynomial time [16], and
3. FDs is possible in linear time [5].

Identifying when INDs and FDs do not interact reduces their combined implication problem to separate, simpler implication problems. For INDs and FDs over null-free relations, conditions for non-interaction were given in [40, 41, 47]. In this section, we will establish a liberal sufficient condition for the non-interaction for FDs and acyclic NNINDs, similar to the condition given in [12] for non-interaction of keys and acyclic NNINDs, and develop a chase procedure for deciding implication for FDs and acyclic NNINDs along the way.

Since every NNIND is logically equivalent to a set of quasi-simple NNINDs by Theorem 6, we may assume that all NNINDs are quasi-simple. With null markers present we need to specify the semantics of functional dependencies. Our definition matches that of [3, 32, 33, 43], and that of *possible FD* in [36].

Definition 13 (functional dependency)**.**
We say that a functional dependency $R : X \to Y$ holds on relation r over relation schema R if and only if for every pair of X-total tuples $t, t' \in r$ with $t[X] = t'[X]$ we have $t[Y] = t'[Y]$.

We illustrate the semantics of FDs from the previous definition on our running example.

Example 21. *Extending Example 1, we may add the following FDs to the given set Σ of INDs:*

- Booking : *CName, PName $\rightarrow$ Dates, Room, and*
- Talk : *Pname, Dates $\rightarrow$ CName.*

The following Booking*-relation satisfies the FD on* Booking *above:*

Booking			
CName	*PName*	*Dates*	*Room*
WWW	$\perp$	*18-22 May 2015*	*VIP 1*
WWW	*Tim*	*18-22 May 2015*	*VIP 2*

The FD above is satisfied as the first tuples is not {CName,PName}-total. However, the relation violates the FD CName, Dates $\rightarrow$ PName, as both tuples are {CName,Dates}-total but have non-matching information on PName.

As FDs and NNINDs reduce to "ordinary" FDs and IND when all attributes are declared not null, their combined finite and unrestricted implication problems include the null-free case, which is already infeasible [49, 17].

Theorem 11. *The finite and unrestricted implication problems for FDs and NNINDs are different and both undecidable.*

Our definition for 'no interaction' matches that of [47] for the null-free case.

Definition 14 (no interaction)**.**
Let Σ_{FD} and Σ_{IND} be sets of FDs and NNINDs over some database schema $\mathcal{R}$, respectively. We say that Σ_{FD} and Σ_{IND} have no interaction*, if and only if for each FD τ and each NNIND φ over $\mathcal{R}$*

$$\Sigma_{FD} \cup \Sigma_{IND} \models \tau \Leftrightarrow \Sigma_{FD} \models \tau \qquad \text{and} \qquad \Sigma_{FD} \cup \Sigma_{IND} \models \varphi \Leftrightarrow \Sigma_{IND} \models \varphi \ .$$

Over null-free relations, being *reduced* is a necessary condition for no interaction [40, 41]. Sets $\Sigma_{\mathtt{FD}}$ and $\Sigma_{\mathtt{IND}}$ of FDs and INDs are reduced if and only if for every IND $R[X] \subseteq S[Y] \in \Sigma_{\mathtt{IND}}$, Y contains no non-trivial FD implied by $\Sigma_{\mathtt{FD}}$. However, being reduced (and acyclic) is not sufficient - see [40] for an example where reduced sets of FDs and acyclic INDs interact. We strengthen this notion into a condition that is sufficient (but not necessary) for acyclic NNINDs. Our definition resembles that of *non-key-conflicting* [10, 11, 12], for which separability results for inclusion dependencies and keys have been established.

Definition 15 (super-reduced)**.**
Let Σ_{FD} and Σ_{IND} be sets of FDs and NNINDs. We call $(\Sigma_{FD}, \Sigma_{IND})$ super-reduced *if and only if for every non-trivial NNIND $R^U[X] \subseteq S^V[Y]$ in Σ_{IND}, VY does not properly contain the LHS Z of any non-trivial FD $S : Z \rightarrow A$ in Σ_{FD} (i.e., we do not have $Z \subsetneq VY$).*

The following example illustrates this definition.

Example 22. *Let Σ_{FD} and Σ_{IND} denote the sets of FDs and NNINDs from Examples 1 and 21. Then the pair $(\Sigma_{FD}, \Sigma_{IND})$ is indeed super-reduced, because the set {CName,PName} of the NNIND*

$$\text{TALK}^{\emptyset}[CName, PName] \subseteq \text{BOOKING}^{\emptyset}[CName, PName]$$

does not properly *contain the LHS of the FD* BOOKING*: CName, PName $\to$ Dates, Room. However, if we added the FD*

$$\text{CHAIR}: CName \to PName$$

to Σ_{FD}, then $(\Sigma_{FD}, \Sigma_{IND})$ would not be super-reduced: For the NNIND

$$\text{BOOKING}^{CName,PName}[CName, PName] \subseteq \text{CHAIR}^{\emptyset}[CName, PName],$$

the set {CName, PName} does properly contain the LHS {CName} of the FD CHAIR*: CName $\to$ PName.*

Note that the condition for being super-reduced could be weakened by replacing VY with $(V \setminus \text{NN}(S)) \cup Y$, since $R^U[X] \subseteq S^V[Y]$ and $R^U[X] \subseteq S^{V\setminus\text{NN}(S)}[Y]$ are equivalent. While this would make the condition independent of the choice of covers for $\Sigma_{\texttt{FD}}$ and $\Sigma_{\texttt{IND}}$, we opted for readability. However, the condition could not be weakened by omitting the set V completely, as the following example demonstrates.

Example 23. *Consider the sets $\Sigma_{FD} = \{S : A \to BC\}$ and $\Sigma_{IND} = \{R^A[A] \subseteq S^B[A],\ R^A[A] \subseteq S^C[A],\ S^{ABC}[A] \subseteq T^{\emptyset}[A]\}$. The set A appearing as the RHS of NNINDs in Σ_{IND} does not properly include the LHS of a FD in Σ_{FD}. However, $\Sigma_{FD} \cup \Sigma_{IND}$ implies $R^A[A] \subseteq T^{\emptyset}[A]$, while Σ_{IND} alone does not.*

Our main goal for this section is to show the following:

Theorem 12. *Let Σ_{FD} and Σ_{IND} be super-reduced sets of FDs and acyclic NNINDs. Then Σ_{FD} and Σ_{IND} have no interaction.*

Our proof of Theorem 12 follows that of [47, Section 10.10]. We first introduce a chase procedure and prove it correct, which is interesting in its own right. By showing the non-applicability of chase rules for super-reduced $\Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}}$, we then prove non-interaction.

Algorithm 2 (Chase for NNINDs and FDs)**.**
Input: *A database schema $(\mathcal{R}, \Sigma_{FD}, \Sigma_{IND})$ with $\mathcal{R} = (R_1, \ldots, R_k)$ and a finite database $\boldsymbol{r} = (r_1, \ldots, r_k)$ over $\mathcal{R}$.*
Output: *A modified database chase$(\boldsymbol{r}, \Sigma_{FD}, \Sigma_{IND})$.*
Method: *Apply the following rules as long as possible:*

NNIND: If Σ_{IND} contains $R^U[X] \subseteq S^V[Y]$ and for some U-total tuple t_R over R there does not exist a V-total tuple t_S over S with $t_R[X] \sqsubseteq t_S[Y]$, then *add the following tuple t_S over S:*

$$t_S[B_i \in Y] = \begin{cases} * & \textit{if } t_R[A_i] = \bot \textit{ and } B_i \in V \cup \mathrm{NN}(S) \\ t_R[A_i] & \textit{otherwise} \end{cases}$$

$$t_S[B \notin Y] = \begin{cases} * & \textit{if } B \in V \cup \mathrm{NN}(S) \\ \bot & \textit{otherwise} \end{cases}$$

where each $*$ *denotes a unique value* $\neq \bot$ *greater than all current values occurring in* $\boldsymbol{r}$*, and* $A_i = (Y \mapsto X)(B_i)$*, as in Algorithm 1.*

FD: If Σ_{FD} *contains a dependency* $R_i : X \to Y$ *and for some* X*-total rows* $t, t' \in r_i$ *it holds that* $t[X] = t'[X]$ *but* $t[A] \neq t'[A]$ *for some* $A \in Y$*, then*

a) if $t[A] = \bot$ *change* $t[A]$ *to* $t'[A]$*, or*

b) if $t'[A] = \bot$ *change* $t'[A]$ *to* $t[A]$*, or*

c) otherwise change each occurrence (in any of the relations of $\boldsymbol{r}$*) of the larger of the values* $t[A]$ *and* $t'[A]$ *to be equal to the smaller one.*

Next we show termination of Algorithm 2 for any given set of FDs and acyclic NNINDs, and prove that its output instance satisfies the input constraints. Note that Algorithm 2 does not require $(\Sigma_{\mathtt{FD}}, \Sigma_{\mathtt{IND}})$ to be super-reduced, and that acyclicity of $\Sigma_{\mathtt{IND}}$ is sufficient but not necessary for termination.

Theorem 13. *The following hold:*

(i) If Σ_{IND} *is acyclic, then Algorithm 2 terminates.*

(ii) If Algorithm 2 terminates, chase$(\boldsymbol{r}, \Sigma_{FD}, \Sigma_{IND})$ *satisfies* Σ_{FD} *and* Σ_{IND}*.*

Proof. As for the null-free case [47], omitted. □

The following theorem shows how the implication problem for FDs and an acyclic set of NNINDs can be decided with the help of the Chase procedure in Algorithm 2.

Theorem 14. *Let* Σ_{IND} *be acyclic,* $\varphi = R^{UX}[X] \subseteq S^V[Y]$*, and* $\boldsymbol{r}$ *consist of a single tuple* t_0 *over* R *with*

(i) $t_0[UX \cup \mathrm{NN}(R)]$ *consisting of pairwise distinct values different from* $\bot$

(ii) $t_0[R - UX \cup \mathrm{NN}(R)] = (\bot, \ldots, \bot)$

Then $\Sigma_{FD} \cup \Sigma_{IND}$ *(finitely) implies* φ *if and only if* φ *holds on chase*$(\boldsymbol{r}, \Sigma_{FD}, \Sigma_{IND})$*.*

Proof. The 'only if' direction follows from Theorem 13. The counter example constructed is finite, which covers the case of finite implication.

For the 'if' direction we begin with some terminology. As we will consider value substitutions, we shall refer to databases prior to substitution as tableaux, and values occurring in them as variables. Denote the universal domain by $\mathcal{U}$, and the variables occurring in a tableaux $\mathbf{r}$ by $var(\mathbf{r})$. An *instantiation* of a set of variables $\mathcal{X}$ is a mapping $\mathcal{X} \to \mathcal{U}$. For any tableaux $\mathbf{r}$ (varying during the chase), an instantiation $\mathcal{I} : var(\mathbf{r}) \to \mathcal{U}$ induces a set $\mathcal{I}(\mathbf{r})$ of *possible instances* of $\mathbf{r}$, each derived by replacing in $\mathbf{r}$ any variable v by $\mathcal{I}(v)$, and any occurrence of the $\perp$ value by some arbitrary value in $\mathcal{U} \cup \{\perp\}$. The same terminology will be used for tuples.

Now let $\mathbf{r}$ denote our tableaux at any point during the chase. Let $\mathcal{I}_t$ be some instantiation of $var(t)$, $t_\mathcal{I}$ a possible instance of t w.r.t. $\mathcal{I}_t$, and $\mathbf{D}$ some database over $\mathcal{R}$ containing $t_\mathcal{I}$ and satisfying $\Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}}$. We claim that $\mathcal{I}_t$ can be extended to an instantiation $\mathcal{I}_\mathbf{r}$ so that for some possible instance $\mathbf{d} \in \mathcal{I}_\mathbf{r}(\mathbf{r})$ we have $\mathbf{d} \subseteq \mathbf{D}$. We show this claim by induction on $\mathbf{r}$. The base case is trivial with $\mathbf{d}$ containing precisely $t_\mathcal{I}$, so assume it holds for $\mathbf{r}$ and denote by $\mathbf{r}'$ the tableaux after a single chase step.

(NNIND) Let the `NNIND` rule be applied, for some NNIND $\varphi' = {R_i}^{U'}[X'] \subseteq {R_j}^{V'}[Y']$. Denote by u the tuple to which the chase rule is applied, by $u_\mathcal{I} \in \mathbf{d}$ its instantiation, and by $u' \in \mathbf{r}'$ the tuple added using the chase rule. Since $\mathbf{D}$ satisfies $\varphi' \in I$ there must exist some tuple $u'_\mathcal{I} \in \mathbf{D}$ which is $V' \cup \mathrm{NN}(R_j)$-total and satisfies $u_\mathcal{I}[X'] \sqsubseteq u'_\mathcal{I}[Y']$. By construction, u' contains new variables or $\perp$ for any attribute outside of Y', and for any attribute A with $u'_\mathcal{I}[A] = \perp$ we also have $u'[A] = \perp$. Hence $\mathcal{I}_\mathbf{r}$ can be extended to $\mathcal{I}_{\mathbf{r}'}$ (by mapping $u'[A]$ to $u'_\mathcal{I}[A]$) so that $u'_\mathcal{I}$ is a possible instance of u' w.r.t. $\mathcal{I}_{\mathbf{r}'}$. Adding $u'_\mathcal{I}$ to $\mathbf{d}$ thus leads to a possible instance $\mathbf{d}'$ of $\mathbf{r}'$ w.r.t. $\mathcal{I}_{\mathbf{r}'}$ with $\mathbf{d}' \subseteq \mathbf{D}$, as claimed.

(FD) Let the `FD` rule be applied, for some FD $R_i : X' \to A \in \Sigma_{\texttt{FD}}$. Denote by u, v the tuples to which the rule is applied, with instantiations $u_\mathcal{I}, v_\mathcal{I} \in d$. Then $\perp \notin u[X'] = v[X']$ and thus $\perp \notin u_\mathcal{I}[X'] = v_\mathcal{I}[X']$. Since $d \subseteq D$ satisfies $\Sigma_{\texttt{FD}}$, we must have $u_\mathcal{I}[A] = v_\mathcal{I}[A] \neq \perp$, with the last inequality following from $u[A] \neq v[A]$ (recall that only $\perp$ can be mapped to $\perp$). If case a) or b) of the `FD` rule applies, i.e., $u[A] = \perp$ or $v[A] = \perp$, then $\mathcal{I}_\mathbf{r} = \mathcal{I}_{\mathbf{r}'}$ and $\mathbf{d}$ is a possible instance of $\mathbf{r}'$. If case c) of the `FD` rule applies, then we must have $\mathcal{I}_\mathbf{r}(u[A]) = \mathcal{I}_\mathbf{r}(v[A])$, so $\mathbf{d}$ is again a possible instance of $\mathbf{r}'$.

This shows our claim about the existence of $\mathbf{d}$. Now assume φ is not implied by $\Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}}$. Then there must exist a database $\mathbf{D}$ satisfying $\Sigma_{\texttt{FD}}$ and $\Sigma_{\texttt{IND}}$ but violating φ. Hence $\mathbf{D}$ contains a UX-total tuple $t_\mathcal{I}$ over R, but no V-total tuple $s_\mathcal{I}$ over S with $t_\mathcal{I}[X] = s_\mathcal{I}[Y]$. Since $t_\mathcal{I}$ is total on $UX \cup \mathrm{NN}(R_i)$, $t_\mathcal{I}$ is a possible instance of t w.r.t. some instantiation $\mathcal{I}_t$, by definition of t. By our claim shown earlier, there exists a database $\mathbf{d} \subseteq \mathbf{D}$ which is a possible instance of $\mathrm{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$ w.r.t. some extension $\mathcal{I}_{\mathrm{chase}}$ of $\mathcal{I}_t$. Now, if $\mathrm{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$ were to contain some V-total tuple s on R_j with $t[X] = s[Y]$, then $\mathbf{d}$ would contain some possible instance $s_\mathcal{I}$ of s w.r.t. $\mathcal{I}_{\mathrm{chase}}$. But then it would follow that $t_\mathcal{I}[X] = s_\mathcal{I}[Y]$, and $s_\mathcal{I}$ is V-total since s is. Since $\mathbf{D}$ contains no such $s_\mathcal{I}$, $\mathrm{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$ contains no such tuple s, and hence violates φ. $\square$

As a corollary of Theorem 14 we obtain the following property for sets of FDs and acyclic NNINDs.

Theorem 15. *Finite and unrestricted implication for sets of FDs and acyclic NNINDs coincide and are decidable in exponential time.*

Proof. Both implication problems coincide because the Chase via Algorithm 2 terminates according to Theorem 13.

The exponential upper bound follows from the facts that the longest chain of NNINDs is bounded by the number of the given relation schemata, and that every tuple in a given R_i-relation can generate at most k tuples in an R_j-relation, where k denotes the number of different NNINDs from R_i to R_j. □

The exponential upper bound of Theorem 15 cannot be improved because of the exponential lower bound already known for the null-free case [23].

We illustrate the application of Theorem 14 and Algorithm 2 to substantiate the claims of Example 23.

Example 24. *Consider again the sets $\Sigma_{FD} = \{\sigma_0 = S : A \rightarrow BC\}$ and $\Sigma_{IND} = \{\sigma_1 = R^A[A] \subseteq S^B[A],\ \sigma_2 = R^A[A] \subseteq S^C[A],\ \sigma_3 = S^{ABC}[A] \subseteq T^{\emptyset}[A]\}$ from Example 23 on the relation schemata $R = \{A\}$, $S = \{A, B, C, D[\text{NN}]\}$, and $T = \{A\}$; in particular, Σ_{IND} is acyclic. In Example 23 we claimed that i) $\Sigma_{FD} \cup \Sigma_{IND}$ implies $R^A[A] \subseteq T^{\emptyset}[A]$, while ii) Σ_{IND} alone does not imply $R^A[A] \subseteq T^{\emptyset}[A]$. We now prove the statements i) and ii) by applying Theorem 14 to them. For statement i) we start off with the following database instance.*

R
A
1

S			
A	B	C	D

T
A

According to Algorithm 2, we apply σ_1 and σ_2 to obtain the following revised instance.

R
A
1

S			
A	B	C	D
1	2	$\perp$	3
1	$\perp$	4	5

T
A

Next, we apply the FD σ_0 to obtain the following instance.

R
A
1

S			
A	B	C	D
1	2	4	3
1	2	4	5

T
A

Finally, we apply σ_3 to obtain the chased database instance chase$(\mathbf{r}, \Sigma_{FD}, \Sigma_{IND})$.

R
A
1

S			
A	B	C	D
1	2	4	3
1	2	4	5

T
A
1

The instance chase$(\mathbf{r}, \Sigma_{FD}, \Sigma_{IND})$ *satisfies* $R^A[A] \subseteq T^{\emptyset}[A]$, *which is therefore implied by* $\Sigma_{FD} \cup \Sigma_{IND}$ *according to Theorem 14.*

For statement ii) the starting point is the same as before, namely the following database instance.

R
A
1

S			
A	*B*	*C*	*D*

T
A

According to Algorithm 2, we apply σ_1 *and* σ_2 *to obtain the following revised instance.*

R
A
1

S			
A	*B*	*C*	*D*
1	*2*	$\perp$	*3*
1	$\perp$	*4*	*5*

T
A

Note that the FD σ_0 *is not part of the input now. We cannot apply the NNIND* σ_3 *because neither of the two S-tuples is ABC-total. Consequently, the last instance denotes chase*$(\mathbf{r}, \emptyset, \Sigma_{IND})$, *which violates* $R^A[A] \subseteq T^{\emptyset}[A]$. *According to Theorem 14,* $R^A[A] \subseteq T^{\emptyset}[A]$ *is not implied by* Σ_{IND} *alone.*

We continue our work towards the non-interaction result of Theorem 12.

Lemma 8. *Let* Σ_{FD} *and* Σ_{IND} *be super-reduced sets of FDs and acyclic NNINDs. Suppose the initial database* $\boldsymbol{r}$ *satisfies the dependencies in* Σ_{FD}. *Then*

(i) in computing chase$(\boldsymbol{r}, \Sigma_{FD}, \Sigma_{IND})$ *the* `FD` *rule is not applied, and*

(ii) $\boldsymbol{r} \subseteq$ *chase*$(\boldsymbol{r}, \Sigma_{FD}, \Sigma_{IND})$.

Proof. To show (i), assume the contrary, and consider the first application of the `FD` rule, for some FD $S : Z \to A \in \Sigma_{\mathtt{FD}}$. Let t_S, t'_S be the two distinct tuples over S with $\perp \notin t_S[Z] = t'_S[Z]$ and $t_S[A] \neq t'_S[A]$. As the original relation satisfies $\Sigma_{\mathtt{FD}}$, at least one of t_S, t'_S must have been added by an application of the `NNIND` rule.

Let t_S be the one added later, using $\varphi = R^{UX}[X] \subseteq S^V[Y] \in \Sigma_{\mathtt{IND}}$ and some UX-total tuple t_R over R. By construction, values outside of Y are unique or $\perp$ by construction at the time of adding t_S, and thus distinct from values in t'_S. Hence we must have $Z \subseteq Y$, and since $(\Sigma_{\mathtt{FD}}, \Sigma_{\mathtt{IND}})$ are super-reduced, VY cannot properly contain Z, so we must have $Z = Y = VY$. But then φ was not violated by t_R due to t'_S over S with $t_R[X] = t_S[Y] = t'_S[Y]$ and $\perp \notin t'_S[V]$, so the `NNIND` rule could not have been applied.

The inclusion $\mathbf{r} \subseteq \text{chase}(\mathbf{r}, \Sigma_{\mathtt{FD}}, \Sigma_{\mathtt{IND}})$ follows immediately, as rule `NNIND` does not modify or remove existing tuples. □

We are now ready to show Theorem 12.

Proof of Theorem 12. Consider an FD τ not implied by $\Sigma_{\texttt{FD}}$, and let $\mathbf{r}$ be a database satisfying $\Sigma_{\texttt{FD}}$ but not τ. By Lemma 8 we have $\mathbf{r} \subseteq \text{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$. Thus $\text{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$ satisfies $\Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}}$ but violates τ, showing $\Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}} \not\models \tau$.

Let then $\varphi = R^{UX}[X] \subseteq S^V[Y]$ be an arbitrary NNIND, and $\mathbf{r}$ be a database containing a single tuple t_0 over R of the form described in Theorem 14. Then by Lemma 8 no FDs are applied in computing $\text{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}})$, so we have $\text{chase}(\mathbf{r}, \Sigma_{\texttt{FD}}, \Sigma_{\texttt{IND}}) = \text{chase}(\mathbf{r}, \emptyset, \Sigma_{\texttt{IND}})$. By Theorem 14 this means $\Sigma_{\texttt{FD}} \cup \Sigma_{\texttt{IND}}$ implies φ iff $\Sigma_{\texttt{IND}}$ alone implies φ. $\square$

We remark that although we assumed all NNINDs to be quasi-simple, a set $\Sigma_{\texttt{IND}}$ of general NNINDs is acyclic and super-reduced, with respect to some set of FDs $\Sigma_{\texttt{FD}}$, if and only if the equivalent set $\Sigma'_{\texttt{IND}}$ of quasi-simple NNINDs, as obtained by Theorem 6, is acyclic and super-reduced.

For acyclicity recall Definition 10: $\Sigma_{\texttt{IND}}$ and $\Sigma'_{\texttt{IND}}$ induce the same directed graph, so either both are acyclic or neither. For being super-reduced, recall Definition 16: VY must not properly contain the LHS of any non-trivial FD in $\Sigma_{\texttt{FD}}$. For each NNIND $R^U[X] \subseteq S^V[Y]$ in $\Sigma_{\texttt{IND}}$ the equivalent quasi-simple NNINDs $R^{UX'}[X'] \subseteq S^V[Y']$ in $\Sigma'_{\texttt{IND}}$ cover all cases of $Y' \subseteq Y$, so either the set VY properly contains the LHS of a FD, or none of its subsets does.

8. Results for null-free relations

In this section we review some of our results for the important special case of relational databases not permitting null values. Firstly, our sufficient condition for the non-interaction between NNINDs and FDs improves on criteria in [47]. While not surprising in light of similar separability results for INDs and keys [10, 11, 12], it is worth mentioning. Secondly, we remark that the fixed parameter-tractability of NNINDs in their arity is already a new result in the case of null-free relations.

8.1. No interaction between FDs and INDs

Studying the non-interaction between FDs and INDs for null-free relations has the same motivation as before. In general, the (finite) implication problem is undecidable, but acyclic INDs by themselves can be decided in NP-time, while implication of FDs can be decided in linear time.

Our results about the non-interaction between NNINDs and FDs reduce to the following definition and theorem for null-free relations.

Definition 16 (super-reduced)**.**
Let Σ_{FD} and Σ_{IND} be sets of FDs and INDs. We call $(\Sigma_{FD}, \Sigma_{IND})$ super-reduced *if and only if for every non-trivial IND $R[X] \subseteq S[Y]$ in Σ_{IND}, Y does not properly contain the LHS Z of any non-trivial FD $S : Z \to A$ in Σ_{FD} (i.e., we do not have $Z \subsetneq Y$).*

Theorem 16. *Let Σ_{FD} and Σ_{IND} be super-reduced sets of FDs and acyclic INDs. Then Σ_{FD} and Σ_{IND} have no interaction.*

Note that Theorem 16 includes both cases mentioned by Mannila and Räihä in [47]: unary inclusion dependencies[6] (Theorem 10.20), and key-based inclusion dependencies on schemata in Boyce-Codd normal form (Theorem 10.21).

Note further that Levene and Loizou [40, 41] consider a stricter definition of 'no interaction', requiring that no subsets of $\Sigma_{\mathtt{FD}}$ and $\Sigma_{\mathtt{IND}}$ interact. Since subsets of super-reduced sets are again super-reduced (and dito for acyclicity), Theorems 12 and 16 still hold with respect to 'no subset interaction'. The same argument could be made for unary inclusion dependencies, but not for key-based INDs on schemata in Boyce-Codd normal form.

8.2. INDs with Bounded Arity

It is well-known [15, 22] that implication for inclusion dependencies of bounded arity can be decided in PTIME. However, Theorem 9 has established the stronger result that NNINDs are fixed parameter-tractable in their arity. This is new, already in the special case of null-free relations.

Theorem 17. *The implication problem for inclusion dependencies is fixed parameter-tractable in their arity.*

9. Conclusion and Future Work

SQL will continue to dominate database practice in the foreseeable future. Domain, entity and referential integrity are therefore fundamental rules that important real-world data is and will be governed by.

Motivated by this, we have studied inclusion dependencies under null marker occurrences, not null constraints, simple and partial semantics. We showed the surprising result that simple INDs and not null constraints together cannot be finitely axiomatized. This prompts the challenging question whether INDs can be defined, in the context of SQL-like databases, such that simple and partial semantics can both be expressed and the good computational properties of INDs from null-free relations are retained. We demonstrated that not null inclusion dependencies (NNINDs) provide an affirmative answer. Indeed, we have established a finite axiomatization, a chase procedure for their implication problem, and that this implication problem is PSPACE-complete. We also showed that implication of NNINDs is fixed parameter-tractable in their arity, and that implication for typed acyclic NNINDs is NP-hard.

Finally, we investigated the interaction of NNINDs with FDs, whose finite and unrestricted implication problems are each undecidable. Finite and unrestricted implication coincide for the combined class of FDs and acyclic NNINDs, and can be decided in exponential time, which we showed by introducing a second chase procedure. In addition, we established a liberal sufficient condition that guarantees the non-interaction of functional dependencies and an acyclic set of (not null) INDs, thereby simplifying computation. The liberal condition

[6]With standard FDs, i.e., for $Z \to A$, Z must be non-empty.

is also effective as it can be verified in quadratic time. Our results provide first insight how to efficiently manage Codd's *combined* set of rules for domain, entity and referential integrity in SQL.

We already mentioned some open problems that warrant future research, but it is worth mentioning additional problems. Mitchell established an elegant axiomatization for the finite implication of FDs and INDs [49, 50] that is not k-ary for any k. Finite and unrestricted implication of FDs and unary INDs are different problems but each axiomatizable and decidable in PTIME [22]. A seminal paper in query optimization is [34], which exploited a chase for FDs and INDs to characterize containment between conjunctive queries. A seminal paper in schema design is [42] which provides justifications for the inclusion dependency normal form over relational databases. These results would all be interesting to investigate in the real-world context of SQL. It would also be interesting to combine NNINDs with the possible and certain keys introduced in [37], and with the possible and certain FD studied in [36].

Acknowledgement

This research is supported by the Marsden Fund Council from New Zealand Government funding, and by the Natural Science Foundation of China (Grant No. 61472263).

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] P. Atzeni and N. M. Morfuni. Functional dependencies in relations with null values. *Inf. Process. Lett.*, 18(4):233–238, 1984.

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

[4] J. Bauckmann, Z. Abedjan, U. Leser, H. Müller, and F. Naumann. Discovering conditional inclusion dependencies. In *CIKM*, pages 2094–2098, 2012.

[5] 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.

[6] J. Biskup and P. A. Bonatti. Controlled query evaluation for enforcing confidentiality in complete information systems. *Int. J. Inf. Sec.*, 3(1):14–27, 2004.

[7] J. Biskup, U. Dayal, and P. A. Bernstein. Synthesizing independent database schemas. In *SIGMOD*, pages 143–151, 1979.

[8] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.

[9] L. Cabibbo. On keys, foreign keys and nullable attributes in relational mapping systems. In *EDBT*, pages 263–274, 2009.

[10] A. Calì, D. Calvanese, and M. Lenzerini. Data integration under integrity constraints. In *Seminal Contributions to Information Systems Engineering*, pages 335–352. Springer, 2013.

[11] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS*, pages 77–86, 2009.

[12] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS*, pages 260–271, 2003.

[13] D. Calvanese, W. Fischl, R. Pichler, E. Sallinger, and M. Simkus. Capturing relational schemas and functional dependencies in RDFS. In *AAAI*, pages 1003–1011, 2014.

[14] D. Calvanese and R. Rosati. Anwering recursive queries under keys and foreign keys is undecidable. In *KRDB*, 2003.

[15] M. Casanova, R. Fagin, and C. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *J. Comput. Syst. Sci.*, 28(1):29–59, 1984.

[16] M. Casanova and V. Vidal. Towards a sound view integration methodology. In *PODS*, pages 36–47, 1983.

[17] A. Chandra and M. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. Comput.*, 14(3):671–677, 1985.

[18] P. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[19] Z. Chen, V. Narasayya, and S. Chaudhuri. Fast foreign-key detection in Microsoft SQL Server PowerPivot for Excel. *PVLDB*, 7(13):1417–1428, 2014.

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

[21] S. Cosmadakis and P. Kanellakis. Functional and inclusion dependencies: A graph theoretic approach. In *PODS*, pages 29–37, 1984.

[22] S. Cosmadakis, P. Kanellakis, and M. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *J. ACM*, 37(1):15–46, 1990.

[23] S. S. Cosmadakis and P. C. Kanellakis. Equational theories and database constraints. In *STOC*, pages 273–284, 1985.

[24] C. J. Date. Referential integrity. In *VLDB*, pages 2–12, 1981.

[25] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[26] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977.

[27] C. Farkas and S. Jajodia. The inference problem: A survey. *SIGKDD Explorations*, 4(2):6–11, 2002.

[28] G. Gottlob, R. Pichler, and F. Wei. Tractable database design and datalog abduction through bounded treewidth. *Inf. Syst.*, 35(3):278–298, 2010.

[29] J. Gryz. Query folding with inclusion dependencies. In *ICDE*, pages 126–133, 1998.

[30] T. A. Halpin and T. Morgan. *Information modeling and relational databases (2. ed.)*. Morgan Kaufmann, 2008.

[31] T. Härder and J. Reinert. Access path support for referential integrity in SQL2. *VLDB J.*, 5(3):196–214, 1996.

[32] S. Hartmann, M. Kirchberg, and S. Link. Design by example for SQL table definitions with functional dependencies. *VLDB J.*, 21(1):121–144, 2012.

[33] S. Hartmann and S. Link. The implication problem of data dependencies over SQL table definitions. *ACM Trans. Database Syst.*, 37(2):13, 2012.

[34] D. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *JCSS*, 28(1):167–189, 1984.

[35] H. Köhler and S. Link. Inclusion dependencies reloaded. In *CIKM*, pages 1361–1370, 2015.

[36] H. Köhler and S. Link. SQL schema design: Foundations, normal forms, and normalization. In *SIGMOD*, pages 267–279, 2016.

[37] H. Köhler, S. Link, and X. Zhou. Possible and certain SQL keys. *PVLDB*, 8(11):1118–1129, 2015.

[38] M. Levene and G. Loizou. Null inclusion dependencies in relational databases. *Inf. Comput.*, 136(2):67–108, 1997.

[39] M. Levene and G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theor. Comput. Sci.*, 206(1-2):283–300, 1998.

[40] M. Levene and G. Loizou. How to prevent interaction of functional and inclusion dependencies. *Inf. Process. Lett.*, 71(3-4):115–125, 1999.

[41] M. Levene and G. Loizou. Guaranteeing no interaction between functional dependencies and tree-like inclusion dependencies. *Theor. Comput. Sci.*, 254(1-2):683–690, 2001.

[42] M. Levene and M. Vincent. Justification for inclusion dependency normal form. *IEEE Trans. Knowl. Data Eng.*, 12(2):281–291, 2000.

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

[44] C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, nominals, and concrete domains. *J. Artif. Intell. Res. (JAIR)*, 23:667–726, 2005.

[45] S. Ma, W. Fan, and L. Bravo. Extending inclusion dependencies with conditions. *Theor. Comput. Sci.*, 515:64–95, 2014.

[46] H. Mannila and K. Räihä. Inclusion dependencies in database design. In *ICDE*, pages 713–718, 1986.

[47] H. Mannila and K. Räihä. *The Design of Relational Databases*. Addison-Wesley, 1992.

[48] M. Memari and S. Link. Index design for enforcing partial referential integrity efficiently. In *EDBT*, pages 217–228. OpenProceedings.org, 2015.

[49] J. Mitchell. The implication problem for functional and inclusion dependencies. *Information and Control*, 56(3):154–173, 1983.

[50] J. Mitchell. Inference rules for functional and inclusion dependencies. In *PODS*, pages 58–69, 1983.

[51] C. Molinaro and S. Greco. Polynomial time queries over inconsistent databases with functional dependencies and foreign keys. *Data Knowl. Eng.*, 69(7):709–722, 2010.

[52] R. Pichler, A. Polleres, S. Skritek, and S. Woltran. Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries. *Semantic Web*, 4(4):351–393, 2013.

[53] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[54] E. Sciore. Inclusion dependencies and the universal instance. In *PODS*, pages 48–57, 1983.

[55] D. Toman and G. Weddell. On keys and functional dependencies as first-class citizens in description logics. *J. Autom. Reasoning*, 40(2-3):117–132, 2008.

[56] M. Zhang, M. Hadjieleftheriou, B. Ooi, C. Procopiuc, and D. Srivastava. On multi-column foreign key discovery. *PVLDB*, 3(1):805–814, 2010.