# The Additivity Problem for Data Dependencies in Incomplete Relational Databases

Mark Levene
University College London
Gower Street
London WC1E 6BT, U.K.
email: mlevene@cs.ucl.ac.uk

George Loizou
Birkbeck College
Malet Street
London WC1E 7HX, U.K.
email: george@dcs.bbk.ac.uk

**Abstract**

Functional dependencies (FDs) and inclusion dependencies (INDs) are the most fundamental integrity constraints that arise in practice in relational databases. We introduce null functional dependencies (NFDs) and null inclusion dependencies (NINDs) to cater for the situation when a database is incomplete and thus contains null values. If a NFD or NIND is weakly satisfied in a database, then there exists a possible world of this database in which the NFD or NIND is satisfied in the standard way. Additivity is the property of the equivalence of the weak satisfaction of a set of NFDs and NINDs, $\Sigma$, with the individual weak satisfaction of each member of $\Sigma$ in the said database. We show that in general satisfaction of NFDs and NINDs is not additive. The problem that arises is: under what conditions is weak satisfaction of NFDs and NINDs additive. We solve this problem for the following cases: when $\Sigma$ is a set of NFDs, when $\Sigma$ is a set of unary NINDs and when $\Sigma$ is a set of NFDs and unary NINDs. We show that, when the set of NINDs is unary, checking whether $\Sigma$ is additive can be done in time polynomial in the size of $\Sigma$.

## 1 Introduction

Functional dependencies (FDs) [ULLM88, ATZE93] generalise the notions of *entity integrity* and *keys* [CODD79] and inclusion dependencies (INDs) [MITC83, CASA84] generalise the notions of *referential integrity* and *foreign keys* [CODD79, DATE86]. In this sense FDs and INDs are the most fundamental data dependencies that arise in practice.

Codd [CODD79] suggested the addition to the database domains of an unmarked null value, whose meaning is "value at present unknown", which we denote by *unk*, in order to extend relations so that they can model situations when the information is incomplete. We call such relations, whose tuples may contain the null value *unk*, *incomplete relations* (or simply relations) and we call a set of incomplete relations an *incomplete database* (or simply a database). Following Codd's proposal, incomplete information is represented in SQL by using *unk* as a distinguished null value [DATE93].

The theory of data dependencies in relational databases has been generalised in order to deal with incomplete databases. In particular, FDs have been generalised to *null functional dependencies* (NFDs) [LIEN82, ATZE86, LEVE94a], multivalued dependencies [ULLM88, ATZE93] have been generalised to *null multivalued dependencies* [LIEN82] and join dependencies [ULLM88, ATZE93] have been generalised to *null join dependencies* [LEVE92]. More recently INDs have been generalised to *null inclusion dependencies* (NINDs) [LEVE94b].

A database schema $\mathbf{R}$ is a collection of relation schemas $\{R_1, \ldots, R_n\}$ and each relation schema $R_i$ has an associated sequence of attributes, denoted by schema$(R_i)$. Correspondingly, a database d over $\mathbf{R}$ is a collection $\{r_1, \ldots, r_n\}$ of relations such that each $r_i$ is a relation over $R_i$. We define the semantics of an incomplete database, d, in terms of the *possible worlds* relative to d, which we denote by POSS(d). POSS(d) is the set of all databases that emanate from all possible substitutions of occurrences of *unk* in d by nonnull values in the database domains.

Intuitively, a NFD R : X $\rightarrow$ Y is *weakly satisfied* (or simply satisfied) in a database d over **R**, if there exists a database $d'$ in POSS(d) such that the relation over R in $d'$ satisfies the NFD on using the standard definition of FD satisfaction [ULLM88, ATZE93].

Correspondingly, we now explain the meaning of a NIND R[X] $\subseteq$ S[Y] being *weakly satisfied* (or simply satisfied) in a database. Intuitively, an occurrence of *unk* is *less informative* than an occurrence of a nonnull value, and a tuple $t_1$ over a relation schema $R_1$, with schema($R_1$) = $< A_1, \ldots, A_m >$, is less informative than a tuple $t_2$ over a relation schema $R_2$, with schema($R_2$) = $< B_1, \ldots, B_m >$, written $t_1 \sqsubseteq t_2$, if $\forall$ i $\in$ {1,. . .,m}, $t_1[A_i]$ is less informative than $t_2[B_i]$. The NIND R[X] $\subseteq$ S[Y] is satisfied in a database, d, containing relations r over R and s over S, if $\forall t_1 \in$ r, $\exists t_2 \in$ s such that $t_1$[X] $\sqsubseteq$ $t_2$[Y]. This definition of a NIND can be seen to faithfully generalise the definition of an IND, since it implies that there exists a database in POSS(d) which satisfies the NIND on using the standard definition of IND satisfaction [MITC83, CASA84].

We write d $\models_{\approx} \alpha$ to denote the fact that an incomplete database, d, *weakly satisfies* a NFD or a NIND $\alpha$ (or simply satisfies $\alpha$). We say that d *weakly satisfies* a set of NFDs and NINDs $\Sigma$ (or simply satisfies $\Sigma$), written d $\models_{\approx} \Sigma$, if d satisfies all the NFDs and NINDs in $\Sigma$ and, in addition, $\exists d' \in$ POSS(d) such that $d'$ satisfies all the NFDs and NINDs in $\Sigma$.

Weak satisfaction (or simply satisfaction) is defined in terms of possible worlds. Intuitively, if an incomplete database, d, satisfies a set of data dependencies, say $\Sigma$, then there exists a sequence of updates, each update modifying a null value to a nonnull value, such that the resulting complete database satisfies all the data dependencies $\alpha \in \Sigma$. The problem that arises is that although there may exist such a sequence of updates for every single data dependency $\alpha \in \Sigma$, two or more such sequences may lead to different possible worlds. In particular, there may not exist a single sequence of updates that leads to one possible world that satisfies all of the data dependencies $\alpha$ in $\Sigma$. From the user's point of view, when d $\not\models_{\approx} \Sigma$, it is natural to view such a database, d, as contradictory. Even if d is a true reflection of the current available information the user may still view $\Sigma$ as contradictory. We call this problem the *additivity problem*. The aim of this paper is to determine how $\Sigma$ can be syntactically restricted so that this problem does not arise.

We next give the formal definition of additive satisfaction and present some motivating examples. Satisfaction is said to be *additive* [ATZE93, LEVE94a] with respect to a class of databases, say *DBC*, and a class of sets of NFDs and NINDs, say *DEPC*, whenever: for all databases d in *DBC* and for all sets of NFDs and NINDs $\Sigma$ in *DEPC*, d $\models_{\approx} \Sigma$ if and only if $\forall \alpha \in \Sigma$, d $\models_{\approx} \alpha$.

It is easy to verify that satisfaction is additive with respect to the class of complete databases (i. e. databases without nulls) and the class of sets of NFDs and NINDs. That is, when the information is complete and there is only a single possible database then the additivity problem does not arise.

We next give three motivating examples each exhibiting typical counterexample databases showing that satisfaction is not additive with respect to the class of incomplete databases and a class of sets of NFDs and/or NINDs. The first example shows that satisfaction is not additive with respect to the class of incomplete databases and the class of all sets of NFDs. The second example shows that satisfaction is not additive with respect to the class of incomplete databases and the class of all sets of NINDs. The third example shows that satisfaction is not additive with respect to the class of incomplete databases and the class of all sets of NFDs and NINDs that contain only a single NFD and a single NIND.

**Example 1** Let $d_1 = \{r_1\}$ be a database, where $r_1$ over R is the well-known incomplete relation [ATZE86] shown in Table 1. It can easily be verified that $d_1 \models_{\approx}$ R : A $\rightarrow$ B and $d_1 \models_{\approx}$ R : B $\rightarrow$ C but $d_1 \not\models_{\approx}$ {R: A $\rightarrow$ B, R : B $\rightarrow$ C}. Also, let $d_2 = \{r_2\}$ be a database, where $r_2$ over R is shown in Table 2. It can easily be verified that $d_2 \models_{\approx}$ R : A $\rightarrow$ C and $d_2 \models_{\approx}$ R : B $\rightarrow$ C, but $d_2 \not\models_{\approx}$ {R : A $\rightarrow$ C, R: B $\rightarrow$ C}. Finally, let $d_3 = \{r_3\}$, where $r_3$ over R is shown in Table 3. It can easily be verified that $d_3 \models_{\approx}$ R : B $\rightarrow$ A and $d_3 \models_{\approx}$ R: AC $\rightarrow$ B, but $d_3 \not\models_{\approx}$ {R : B $\rightarrow$ A, R : AC $\rightarrow$ B}.

| A | B | C |
|---|---|---|
| 0 | *unk* | 0 |
| 0 | *unk* | 1 |

Table 1: The counterexample relation $r_1$

| A | B | C |
|---|---|---|
| 0 | *unk* | 0 |
| 0 | 0 | *unk* |
| *unk* | 0 | 1 |

Table 2: The counterexample relation $r_2$

| A | B | C |
|---|---|---|
| 0 | 0 | *unk* |
| *unk* | 0 | 0 |
| 0 | 1 | 0 |

Table 3: The counterexample relation $r_3$

**Example 2** Let r, s and z be the relations shown in Tables 4, 5 and 6, respectively, and let d = {r, s, z} be a database over **R**, with **R** = {R, S, T} and schema(R) = <A>, schema(S) = <B> and schema(T) = <C>. It can easily be verified that d $\models_{\approx}$ R[A] $\subseteq$ S[B] and d $\models_{\approx}$ R[A] $\subseteq$ T[C] but d $\not\models_{\approx}$ {R[A] $\subseteq$ S[B], R[A] $\subseteq$ T[C]}.

| A |
|---|
| *unk* |

Table 4: The counterexample relation r

| B |
|---|
| 0 |

Table 5: The counterexample relation s

| C |
|---|
| 1 |

Table 6: The counterexample relation z

**Example 3** Let r be the relation shown in Table 7 and let d = {r} be a database over **R**, with **R** = {R} and schema(R) = <A, B>. It can easily be verified that d $\models_{\approx}$ A $\rightarrow$ B and d $\models_{\approx}$ R[A] $\subseteq$ R[B] but d $\not\models_{\approx}$ {A $\rightarrow$ B, R[A] $\subseteq$ R[B]}.

| A | B |
|---|---|
| 0 | 0 |
| 1 | 0 |
| *unk* | 1 |

Table 7: A counterexample relation

We are interested in finding the largest class of sets of NFDs and NINDs for which satisfaction is additive. We solve the problem for NFDs and unary NINDs separately, and for NFDs and unary NINDs together. (A NIND R[X] $\subseteq$ S[Y] is *unary* if both X and Y are single attributes [COSM90].) We now briefly describe the solution to the additivity problem.

We show that a set of NFDs, F, is additive if and only if it is *monodependent*. Informally, F is monodependent if for each attribute A there is a unique NFD that functionally determines A, and in addition only trivial cycles involving A arise between any two NFDs one of which functionally determines A (see Definition 6.2).

We also show that a set of unary NINDs, I, is additive if and only if it is *monodirectional*. Informally, I is monodirectional if for every pair of unary NINDs that have a common left-hand side, there exists at least one compensating unary NIND that is logically implied by I (see Definition 7.1).

We further show that a set of NFDs and unary NINDs, $\Sigma$ = F $\cup$ I, is additive if and only if F is monodependent, I is monodirectional and $\Sigma$ satisfies the *cyclicity condition*. Informally, $\Sigma$ satisfies the cyclicity condition if whenever the left-hand side of a NIND is contained in the left-hand side of a NFD or vice versa, then there exist a compensating NFD and a compensating unary NIND that are logically implied by $\Sigma$ (see Definition 8.1).

Finally, we show that as long as the set of NINDs is unary additive satisfaction can be checked in polynomial time in the size of the set of NFDs and NINDs under consideration.

The rest of the paper is organised as follows. In Section 2 we formalise incomplete relations and incomplete databases, and define a partial order on tuples of such relations. In Section 3 we define the notions of NFD and NIND and their satisfaction in a database. In Section 4 we extend the *chase procedure*, defined over complete databases with respect to FDs [MAIE79, HONE82, ATZE93] and INDs [JOHN84, MANN92], to incomplete databases with respect to NFDs and NINDs. In order to establish our results we make extensive use of the chase procedure as a theorem proving tool. In Section 5 we formalise the additivity problem for data dependencies. In Section 6 we solve the additivity problem for NFDs, in Section 7 we solve the additivity problem for unary NINDs, and in Section 8 we solve the additivity problem for NFDs and unary NINDs. In Section 9 we relate our work to previous research pertaining to incomplete information as it affects the satisfaction of data dependencies in relations. Finally, in Section 10 we give our concluding remarks.

# 2 Relations that Model Incomplete Information

In this section we briefly introduce the underlying database model used throughout the paper.

**Definition 2.1 (Relation and database schema)** Let Att be a countably infinite set of attributes. A *relation schema* is a relation symbol R together with an associated similarity type, denoted by type(R), such that type(R) $\in \omega$, where $\omega$ is the set of all natural numbers. We refer to such a schema simply as R. For convenience we associate with each relation schema R a total and one-to-one mapping, att: $\{1, \ldots, \text{type(R)}\} \rightarrow$ Att, which allows us to name the type(R) components of a relation schema R; we denote the sequence $<$att(1), ..., att(type(R))$>$ by schema(R).

A *database schema* is a set $\mathbf{R} = \{R_1, \ldots, R_n\}$, n $\in \omega$, such that each $R_i \in \mathbf{R}$ is a relation schema.

From now on we will refer to a sequence of attributes as a shorthand for a sequence of distinct attributes (we assume that sequences of attributes do not contain any repeated attributes). As usual uppercase letters appearing at the end of the alphabet will be used to denote sequences of attributes, while those at the beginning of the alphabet will be used to denote single attributes. When convenient we will write $A_1, \ldots, A_n$ instead of $< A_1, \ldots, A_n >$ to describe a sequence and when no ambiguity arises we will refer to a sequence of attributes as a set of attributes. We take A $\in < A_1, ..., A_m >$ to mean A $\in \{A_1, ..., A_m\}$ and $< A_1, ..., A_m > \subseteq < B_1, ..., B_k >$, with m $\leq$ k, to mean $\{A_1, ..., A_m\} \subseteq \{B_1, ..., B_k\}$. The difference between two sequences of attributes, denoted by X$-$Y, is the sequence resulting from removing all the common attributes in X and Y from X while maintaining the original order of the attributes remaining in X. The concatenation of two sequences X and Y is denoted by XY; if the sequences X, Y are *not* disjoint we define their concatenation XY to be (X$-$Y)Y.

For a set or a sequence X, |X| will denote the cardinality of X. Finally, the *size* of a set or sequence X is defined to be the cardinality of a standard encoding [GARE79] of X.

**Definition 2.2 (Relation and database)** Let *Dom* be a countably infinite set of values and let $unk, inc \in Dom$ be two distinguished values, denoting the null values “unknown” and “inconsistent”, respectively.

A *type(R)-tuple* (or simply a tuple whenever type(R) is understood from context) is a member of $Dom \times \ldots \times Dom$ (type(R) times).

An *incomplete relation* (or simply a relation) over R is a finite (possibly empty) set of type(R)-tuples. An *incomplete database* (or simply a database) over $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a set d = $\{r_1, \ldots, r_n\}$ such that each $r_i \in$ d is a relation over $R_i \in \mathbf{R}$.

We call values in $Dom - \{unk, inc\}$ nonnull values and the values *unk* and *inc* (unmarked) null values. We let DB($\mathbf{R}$) denote the countably infinite set of all databases over $\mathbf{R}$. From now on we will assume that d = $\{r_1, \ldots, r_n\}$ is a database over a database schema $\mathbf{R} = \{R_1, \ldots, R_n\}$ and that r is a relation over a relation schema R.

**Definition 2.3 (Projection)** The *projection* of a tuple t $\in$ r onto the attribute $A_i$ = att(i), denoted by t[$A_i$], is the i-coordinate of t, i. e. t(i). Projection is extended to a sequence of attributes Y = $< A_1, \ldots, A_m > \subseteq$ schema(R), denoted by t[Y] (also called the Y-value of t), and to relations, denoted by $\pi_Y$(r), in the usual manner [ULLM88, ATZE93].

**Definition 2.4 (Active domain)** The *active domain* of a relation r over R, denoted by adom(r), is defined by

$$adom(r) = \{v \mid \exists A \in schema(R), \exists t \in r \text{ such that } t[A] = v\} \cup \{unk, inc\}.$$

The active domain of a database d = $\{r_1, \ldots, r_n\}$ over **R**, denoted by adom(d), is defined by

$$adom(d) = \bigcup \{adom(r_i) \mid r_i \in d\}.$$

**Definition 2.5 (Complete and consistent relations and databases)** A type(R)-tuple t is said to be *complete* if $\forall A_i \in$ schema(R), t[$A_i$] $\neq$ *unk* and t[$A_i$] $\neq$ *inc*, i. e. t[$A_i$] is a nonnull value, otherwise t is said to be *incomplete*; t is said to be *inconsistent* if $\exists A_i \in$ schema(R), such that t[$A_i$] = *inc*, otherwise t is said to be *consistent.*

A relation r over R is said to be *complete* if $\forall t \in r$, t is complete, otherwise r is said to be *incomplete.* A relation r over R is said to be *inconsistent* if $\exists t \in r$ such that t is inconsistent, otherwise r is said to be *consistent.*

A database d over R is said to be *complete* if all the relations r in d are complete, otherwise d is said to be *incomplete.* A database d over R is said to be *inconsistent* if there exists a relation r in d such that r is inconsistent, otherwise d is said to be *consistent.*

We let COMPLETE(**R**) denote the countably infinite set of all complete databases over **R**.

**Definition 2.6 (Less informative values)** Let r be a relation over R. We define a partial order in *Dom*, denoted by $\sqsubseteq$, as follows: where $u, v \in Dom$,

$$u \sqsubseteq v \text{ if and only if } u = v \text{ or } u = unk \text{ or } v = inc.$$

**Definition 2.7 (Less informative tuples)** Let schema($R_1$) = $< A_1, \ldots, A_m >$ and schema($R_2$) = $< B_1, \ldots, B_m >$, i. e. type($R_1$) = type($R_2$) = m. We extend $\sqsubseteq$ to be a partial order in the set of type($R_1$)-tuples and type($R_2$)-tuples, as follows: where $t_1$ is a type($R_1$)-tuple and $t_2$ is a type($R_2$)-tuple, $t_1$ is *less informative than* $t_2$ (or equivalently $t_2$ is *more informative than* $t_1$), written $t_1 \sqsubseteq t_2$, if

$$\forall i \in \{1, \ldots, m\}, t_1[A_i] \sqsubseteq t_2[B_i].$$

We observe that the set of all type(R)-tuples is a *lattice* having no infinite chains [DAVE90], whose bottom element is $<$*unk*, ..., *unk*$>$ and whose top element is $<$*inc*, ..., *inc*$>$. We next define the *join* operator [DAVE90] of this complete lattice of tuples.

**Definition 2.8 (The join operator)** Let r be a relation over R. We define the *join* operator, denoted by $\sqcup$, as a mapping from an ordered pair, $(v_1, v_2)$, of values in *Dom* $\times$ *Dom* to a single value in *Dom* as follows:

$v_1 \sqcup v_2$ is the least upper bound of $v_1$ and $v_2$ with respect to $\sqsubseteq$.

We extend $\sqcup$ to be a mapping from an ordered pair, $(t_1, t_2)$, of type(R)-tuples to a single type(R)-tuple as follows:

$t_1 \sqcup t_2$ = t, where t is a type(R)-tuple and $\forall A_i \in$ schema(R), t$[A_i] = t_1[A_i] \sqcup t_2[A_i]$.

It can easily be verified that $t_1 \sqcup t_2$ returns the least upper bound of $t_1$ and $t_2$, i. e. the join operator indeed implements the lattice-theoretic join.

**Definition 2.9 (The set of possible worlds of a relation)** The set of all *possible worlds* relative to a relation r over R, denoted by POSS(r), is defined by

$$POSS(r) = \{s \mid s \text{ is a relation over R and there exists a total and onto mapping } f : r \rightarrow s \text{ such that } \forall t \in r, t \sqsubseteq f(t) \text{ and } f(t) \text{ is complete}\}.$$

We extend POSS to a database d = $\{r_1, \ldots, r_n\}$ over **R**, as follows:

$$POSS(d) = \{\{s_1, \ldots, s_n\} \mid s_1 \in POSS(r_1) \text{ and } \ldots \text{ and } s_n \in POSS(r_n)\}.$$

The following proposition states the relationship between the concept of possible worlds and consistency.

**Proposition 2.1** The following two statements are true:

1. A relation r over R is inconsistent if and only if POSS(r) = $\emptyset$.
2. A database d over **R** is inconsistent if and only if POSS(d) = $\emptyset$. $\square$

From now on we assume that unless otherwise stated relations and databases are consistent.

# 3 Null functional and null inclusion dependencies

In this section we introduce the null data dependencies we investigate in the paper.

**Definition 3.1 (Null Functional Dependency)** A *null functional dependency* (or simply a NFD) over a database schema **R** is a statement of the form R : X $\rightarrow$ Y (or simply X $\rightarrow$ Y whenever R is understood from context), where R $\in$ **R** and X, Y $\subseteq$ schema(R).

A NFD X $\rightarrow$ Y is said to be *trivial* if Y $\subseteq$ X; it is said to be *standard* if X $\neq \emptyset$.

Hereafter *we will assume that all NFDs are standard.* We also assume that a database schema **R** has a set of NFDs associated with it, denoted by F. We let F(R) be the subset of NFDs in F over a relation schema R $\in$ **R**, that is F(R) = {R : X $\rightarrow$ Y | R : X $\rightarrow$ Y $\in$ F}. We let NFD(**R**) be the finite set of all possible sets F of NFDs over **R**.

The following definition is a syntactic definition of NFD satisfaction.

**Definition 3.2 (Satisfaction of a NFD)** A NFD R : X $\rightarrow$ Y over **R** is satisfied in a database d, over **R**, denoted by d $\models\!\approx$ R : X $\rightarrow$ Y, if, where $r \in d$ is the relation over R $\in$ **R**, r is consistent and $\forall t_1, t_2 \in$ r, if $t_1$[X] and $t_2$[X] are complete and $t_1$[X] = $t_2$[X], then $t_1[Y] \sqcup t_2[Y]$ is consistent.

From now on we will assume that when R : X $\rightarrow$ Y $\in$ F and d is a database over **R**, then r $\in$ d is the relation over R $\in$ **R**. We observe that if r $\in$ d is a complete relation then d $\models\!\approx$ X $\rightarrow$ Y if and only if the standard definition of functional dependency (FD) is satisfied, i. e. $\forall t_1, t_2 \in$ r, if $t_1$[X] = $t_2$[X], then $t_1$[Y] = $t_2$[Y] [ULLM88, ATZE93].

The following proposition gives a semantic characterisation of NFD satisfaction in terms of possible worlds.

**Proposition 3.1** Let d be a database over $\mathbf{R}$ and $r_1 \in$ d be the relation over $R_1 \in \mathbf{R}$ such that $r_1$ is consistent. Then d $\models\!\approx R_1$ : X $\rightarrow$ Y if and only if $\exists r_2 \in$ POSS($r_1$) such that $\forall t_1, t_2 \in r_2$, if $t_1$[X] = $t_2$[X], then $t_1$[Y] = $t_2$[Y]. $\square$

As a result of Proposition 3.1, whenever appropriate, we assume without loss of generality that the left-hand and right-hand sides of NFDs are disjoint, and moreover that the right-hand sides of NFDs are singletons.

**Definition 3.3 (Null Inclusion Dependency)** A *null inclusion dependency* (or simply a NIND) over a database schema $\mathbf{R}$ is a statement of the form R[X] $\subseteq$ S[Y], where R, S $\in \mathbf{R}$ and X $\subseteq$ schema(R), Y $\subseteq$ schema(S) are sequences of attributes such that $|$X$|$ = $|$Y$|$.

A NIND R[X] $\subseteq$ S[Y] is said to be unary if $|$X$|$ = 1. A NIND is said to be *trivial* if it is of the form R[X] $\subseteq$ R[X].

We assume that a database schema $\mathbf{R}$ has a set of NINDs associated with it, denoted by I, and let NIND($\mathbf{R}$) be the finite set of all possible sets I of NINDs over $\mathbf{R}$.

The following definition is a syntactic definition of NIND satisfaction.

**Definition 3.4 (Satisfaction of a NIND)** A NIND R[X] $\subseteq$ S[Y] over $\mathbf{R}$ is satisfied in a database d over $\mathbf{R}$, denoted by d $\models\!\approx$ R[X] $\subseteq$ S[Y], if d is consistent and $\forall t_1 \in$ r, $\exists t_2 \in$ s such that $t_1$[X] $\sqsubseteq$ $t_2$[Y], where r $\in$ d is the relation over R $\in \mathbf{R}$ and s $\in$ d is the relation over S $\in \mathbf{R}$.

From now on we will assume that when R[X] $\subseteq$ S[Y] $\in$ I and d is a database over $\mathbf{R}$, then r $\in$ d is the relation over R $\in \mathbf{R}$ and s $\in$ d is the relation over S $\in \mathbf{R}$. We observe that if d is a complete database then d $\models\!\approx$ R[X] $\subseteq$ S[Y] if and only if the standard definition of inclusion dependency (IND) is satisfied, i. e. $\pi_X$(r) $\subseteq$ $\pi_Y$(s) [CASA84]. The justification for Definition 3.4 is that the partial order of less informative tuples extended to relations generalises set inclusion in a natural way.

The following proposition gives a semantic characterisation of NIND satisfaction in terms of possible worlds.

**Proposition 3.2** Let $d_1$ be a database over $\mathbf{R}$ and R[X] $\subseteq$ S[Y] be a NIND over $\mathbf{R}$. If $d_1 \models\!\approx$ R[X] $\subseteq$ S[Y] then $\exists d_2 \in$ POSS($d_1$) such that $\pi_X$(r) $\subseteq$ $\pi_Y$(s), where r $\in d_2$ is the relation over R $\in \mathbf{R}$ and s $\in d_2$ is the relation s over S $\in \mathbf{R}$. $\square$

We note that the converse of the above proposition does not hold. For example, let $d_1 = < r_1, r_2 >$ be a database over $\mathbf{R} = \{R_1, R_2\}$, where $r_1$ = {<a, b>} is the relation over $R_1$, with schema($R_1$) = <A, B>, and $r_2$ = {<*unk*, c>} is the relation over $R_2$, with schema($R_2$) = <C, D>. Then $d_1 \not\models\!\approx R_1$[A] $\subseteq R_2$[C] but $d_2 \models\!\approx R_1$[A] $\subseteq R_2$[C], where $d_2 = < r_1, s_2 > \in$ POSS($d_1$) is a database over $\mathbf{R}$, with $s_2$ = {<a, c>} being a relation over $R_2$, and $\pi_{<A>}(r_1) \subseteq \pi_{<C>}(s_2)$. Thus the existence of a possible world of a database, say d, that satisfies an IND is not sufficient for a corresponding NIND to be satisfied in d. Consequently, it is not possible to relax Definition 3.4 so that the converse of Proposition 3.2 would hold, since in that case, in general, referential integrity would be violated.

From now on we will assume that $\Sigma$ = F $\cup$ I is a set of NFDs and NINDs over $\mathbf{R}$. We will also assume without loss of generality that $\Sigma$ does not contain any trivial NFDs or NINDs.

**Definition 3.5 (Logical implication)** We say that $\Sigma$ *logically implies* a NFD or a NIND $\alpha$ over $\mathbf{R}$, written $\Sigma \models\!\approx \alpha$, whenever $\forall$ d $\in$ DB($\mathbf{R}$), the following condition is true:

$$\text{if } \forall \beta \in \Sigma, d \models\!\approx \beta \text{ holds then } d \models\!\approx \alpha \text{ also holds.}$$

The implication problem is to determine the computational complexity of the logical implication $\Sigma \models\approx \alpha$, where as before $\Sigma$ = F $\cup$ I and $\alpha$ is a NFD or a NIND. In the special case where $\Sigma$ = F the implication problem can be solved in linear time in the size of F [BEER79, ATZE86] and in the special case when $\Sigma$ = I the implication problem is PSPACE-complete [CASA84] (in [LEVE94b] it was shown that the implication problems for INDs and NINDs coincide). In the case where $\Sigma$ may contain both NFDs and NINDs the implication problem is decidable and EXPTIME-complete [LEVE94b] (for FDs and INDs it was shown that the implication problem is undecidable [MITC83]).

We close this section by presenting a relevant result pertaining to unary NINDs. We say that a set of NFDs F over **R** and a set of NINDs I over **R** have *no interaction* if for each NFD $\alpha$ over **R** and for each NIND $\beta$ over **R**, F $\cup$ I $\models\approx \alpha$ if and only if F $\models\approx \alpha$ and F $\cup$ I $\models\approx \beta$ if and only if I $\models\approx \beta$.

**Theorem 3.3** Let $\Sigma$ = F $\cup$ I be a set of NFDs and unary NINDs over **R**. Then F and I have no interaction.

*Proof.* The result follows immediately by the soundness and completeness of the axiom system for NFDs and NINDs given in [LEVE94b]. We now present an alternative proof, which does not utilise this result.

Suppose that F $\not\models\approx$ R : X $\rightarrow$ Y. Then $\exists$ d $\in$ DB(**R**) such that $\forall\alpha \in$ F, d $\models\approx \alpha$ but d $\not\models\approx$ R : X $\rightarrow$ Y. The database d can always be made to satisfy all the NINDs in I by adding tuples to s, where s $\in$ d is the relation over S, of the form $\{< unk, \ldots, unk, v, unk, \ldots, unk >\}$, with the B-value being $v$, whenever a unary NIND of the form R[A] $\subseteq$ S[B] is violated in d. Suppose that I $\not\models\approx$ R[A] $\subseteq$ S[B]. Then $\exists$ d $\in$ DB(**R**) such that $\forall\beta \in$ I, d $\models\approx \beta$ but d $\not\models\approx$ R[A] $\subseteq$ S[B]. Let $d'$ be a database consisting of r = $\{< unk, \ldots, unk, 0, unk, \ldots, unk >\}$, with the A-value being 0, s = $\{< unk, \ldots, unk, 1, unk, \ldots, unk >\}$, with the B-value being 1, and all other relations being empty. The database $d'$ can be made to satisfy all the NINDs in I by adding tuples, having at most a single nonnull value, to the relations in $d'$ thus obtaining the required database d. The result follows, since $\forall\alpha \in$ F, d $\models\approx \alpha$ due to the fact that the tuples in d do not have more than one nonnull value. $\square$

The implication problem for unary NINDs can be solved in linear time in the size of the input set of NINDs [COSM90]. On using Theorem 3.3 the next corollary follows.

**Corollary 3.4** Let $\Sigma$ = F $\cup$ I be a set of NFDs and unary NINDs. The implication problem for F and I can be solved in linear time in the size of $\Sigma$. $\square$

## 4 The chase procedure for NFDs and NINDs

Herein we extend the standard chase procedure defined over complete databases with respect to FDs [MAIE79, HONE82, ATZE93] and INDs [JOHN84, MANN92] to incomplete databases with respect to NFDs and NINDs. In order to establish our results in the following sections we make extensive use of the chase procedure as a theorem proving tool.

Let Dum = $\{\perp_1, \perp_2, \ldots, \perp_q\}$ be a set of distinguished nonnull values in *Dom*, which do not appear in d, where q denotes the finite number of distinct occurrences of *unk* in d. For the purpose of defining the chase procedure we extend the partial order in *Dom* as follows: $\forall$ i,j $\in$ {1,. . .,q}, $\perp_i \sqsubseteq \perp_j$, and for all nonnull values $v$ appearing in d, we have that $\forall\perp_i \in$ Dum, $\perp_i \sqsubseteq v$ but $v \not\sqsubseteq \perp_i$. Furthermore, we extend $\sqcup$ as follows: if i $\leq$ j then $\perp_i \sqcup \perp_j = \perp_i$, otherwise $\perp_i \sqcup \perp_j = \perp_j$.

We now define two *chase rules*, which are applied to a database d over **R**.

**NFD rule:** if R : X $\rightarrow$ Y $\in$ F and $\exists t_1, t_2 \in r$ such that $t_1$[X] = $t_2$[X] and $t_1$[Y] $\neq$ $t_2$[Y], then $\forall$ A $\in$ Y, change all occurrences of $t_1$[A] and $t_2$[A] *in d* to $t_1$[A] $\sqcup$ $t_2$[A].

**NIND rule:** if R[X] $\subseteq$ S[Y] $\in$ I, and $\exists t_1 \in$ r, $\exists t_2 \in$ s, with $t_1$[X] $\sqsubseteq$ $t_2$[Y] or $t_2$[X] $\sqsubseteq$ $t_1$[Y], but $\nexists t_3 \in$ s with $t_1$[X] = $t_3$[Y], then $\forall$ i $\in$ {1,. . .,m}, change all occurrences of $t_1[A_i]$ and $t_2[B_i]$ *in d*, to $t_1[A_i] \sqcup t_2[B_i]$, where X = $< A_1 \ldots, A_m >$ and Y = $< B_1 \ldots, B_m >$.

We now give the pseudo-code of an algorithm designated, CHASE(d, F, I), which, given an input d, F and I, applies the NFD and NIND rules to d as long as possible and returns the resulting database over **R**. CHASE(d, F, I) will also denote the database resulting from the ensuing Algorithm 1.

**Algorithm 1** (CHASE(d, F, I))

```
1.  begin
2.    Result := d;
3.    i := 1;
4.    for each r ∈ Result, with r over R, do
5.      for each t ∈ r and A ∈ schema(R) such that t[A] = unk do
6.          t[A] := ⊥_i;
7.          i := i + 1;
8.      end for
9.    end for
10.   Tmp := ∅;
11.   while Tmp ≠ Result do
12.     Tmp := Result;
13.     Apply the NFD rule or the NIND rule to Result;
14.   end while
15.   return Result;
16. end.
```

The following proposition states an important property of the chase procedure [MAIE79, VASS80, HONE82, ATZE93] with respect to NFDs, which will be useful in Section 6. In the sequel we call an execution of line 13 in Algorithm 1 a *chase step* and we say that the chase step *applies* the NFD or NIND rule to the current state of d.

**Proposition 4.1** Let d be a consistent database over **R** and let F be a set of NFDs over **R**. Then $\exists d_1 \in$ POSS(d) such that $\forall$ R : X $\rightarrow$ Y $\in$ F, $d_1 \approx\!\!\!|$ R : X $\rightarrow$ Y if and only if CHASE(d, F, $\emptyset$) is consistent, or equivalently, CHASE(d, F, $\emptyset$) $\in$ POSS(d). $\square$

It can be verified that CHASE(d, F, $\emptyset$) is unique only up to the order in which the values in Dum are assigned to the null attribute values in d (cf. [MAIE79, ATZE93]). This is due to the fact that the for loops beginning at lines 4 and 5 of Algorithm 1 do not specify the order in which these statements are to be executed.

On the other hand, CHASE(d, $\emptyset$, I) is *not* necessarily unique even up to the order in which the values in Dum are assigned to the null attribute values in d. For example, let d = {r, s} be a database, where r over R, with schema(R) = <A>, is the relation shown in Table 8 and s over S, with schema(S) = <B>, is the relation shown in Table 9. Furthermore, let I = {R[A] $\subseteq$ S[B]}. Then the occurrence of $\perp_i$ in r may be changed to 0 or to 1 depending on whether the NIND rule for R[A] $\subseteq$ S[B] chooses the first or second tuple in s.

From now on we will assume that the for loops beginning at lines 4 and 5 of Algorithm 1 assign the values in Dum to the null attribute values in d in some arbitrary but fixed order. Moreover, we will denote the set of all consistent databases resulting from executing Algorithm 1 by SETOF(CHASE(d, F, I)).

| A |
|---|
| $\perp_i$ |

Table 8: The counterexample relation r

| B |
|---|
| 0 |
| 1 |

Table 9: The counterexample relation s

The following proposition can be proved by utilising the same technique used in [AHO79, Theorem 1], since each application of a chase step reduces the number of distinct values in the database by at least one.

**Proposition 4.2** CHASE(d, F, I) can be computed in time polynomial in the sizes of d, F and I. $\Box$

On the negative side SETOF(CHASE(d, F, I)) may contain an exponential number of databases in the sizes of d and I due to the nonuniqueness of CHASE(d, $\emptyset$, I). For example, let d = {r} be a database with schema(R) = < $A_1, A_2, \ldots, A_m$, B, C>, where r is shown in Table 10. Also, let I = $\cup_{i=1}^{m}$ {R[$A_i$] $\subseteq$ R[B], R[$A_i$] $\subseteq$ R[C]} be a set of unary NINDs. It follows that |SETOF(CHASE(d, $\emptyset$, I))| = $2^n$ but the only database that satisfies all the NINDs in I is the database resulting from modifying all the occurrences of $\perp_i$, for $i = 1, \ldots, m$, to 0.

| $A_1$ | $A_2$ | ... | $A_m$ | B | C |
|---|---|---|---|---|---|
| $\perp_1$ | $\perp_2$ | ... | $\perp_m$ | 0 | 0 |
| 0 | 0 | ... | 0 | 1 | 0 |

Table 10: A relation inducing an exponential number of distinct chases

**Proposition 4.3** Let d be a consistent database over **R** and let I be a set of NINDs over **R**. Then $\exists d_1 \in$ POSS(d) such that $\forall$ R[X] $\subseteq$ S[Y] $\in$ I, $d_1 \models_{\approx}$ R[X] $\subseteq$ S[Y] if and only if $\exists d_2 \in$ SETOF(CHASE(d, $\emptyset$, I)) such that $\forall$ R[X] $\subseteq$ S[Y] $\in$ I, $d_2 \models_{\approx}$ R[X] $\subseteq$ S[Y].

*Proof.* (*If.*) Let $d_2 \in$ SETOF(CHASE(d, $\emptyset$, I)) be a database such that $\forall$ R[X] $\subseteq$ S[Y] $\in$ I, $d_2 \models_{\approx}$ R[X] $\subseteq$ S[Y]. Moreover, let us replace all occurrences of $\perp_i \in$ adom($d_2$), where i $\in$ {1,...,q}, by a distinct nonnull value $v_i \in$ *Dom* that does not appear in adom($d_2$), thus obtaining the database $d_1$. The result follows, since $d_1 \in$ POSS(d) and $\forall$ R[X] $\subseteq$ S[Y] $\in$ I, $d_1 \models_{\approx}$ R[X] $\subseteq$ S[Y].

(*Only if.*) Suppose that $\not\exists d_2 \in$ SETOF(CHASE(d, $\emptyset$, I)) such that $\forall$ R[X] $\subseteq$ S[Y] $\in$ I, $d_2 \models_{\approx}$ R[X] $\subseteq$ S[Y]. Equivalently, $\forall d_2 \in$ SETOF(CHASE(d, $\emptyset$, I)), $\exists$ R[X] $\subseteq$ S[Y] $\in$ I such that $d_2 \not\models_{\approx}$ R[X] $\subseteq$ S[Y]. Let $d_2 \in$ SETOF(CHASE(d, $\emptyset$, I)) and R[X] $\subseteq$ S[Y] $\in$ I be a database and a NIND, respectively, such that $d_2 \not\models_{\approx}$ R[X] $\subseteq$ S[Y]. Furthermore, let us replace all occurrences of $\perp_i \in$ adom($d_2$), where i $\in$ {1,...,q}, by a distinct nonnull value $v_i \in$ *Dom* that does not appear in Dum, thus obtaining the database $d_1$. Obviously, $d_1 \not\models_{\approx}$ R[X] $\subseteq$ S[Y] independently of the choice of $v_i$ for each i $\in$ {1,...,q}. Now $d_1$ is an arbitrary element of POSS(d) realising a possible sequence of assignments of the null values in d to nonnull values in *Dom*. The result follows, since this implies that no sequence of assignments of the null values in d to nonnull values in *Dom* is possible for which $\forall$ R[X] $\subseteq$ S[Y] $\in$ I, $d_1 \models_{\approx}$ R[X] $\subseteq$ S[Y] obtains. $\Box$

We next show that the chase for NFDs and NINDs can be computed by first applying the NIND rule as long as possible and then applying the NFD rule as long as possible (cf. [MANN92] for a similar result concerning noncircular sets of INDs, that is sets of nontrivial INDs, which do not logically imply INDs of the form R[X] $\subseteq$ R[Y]).

**Proposition 4.4** Let d be a consistent database over **R** and let $\Sigma = \text{F} \cup \text{I}$ be a set of NFDs and NINDs over **R**. Then $\exists d_1 \in$ SETOF(CHASE(d, F, I)) such that $\forall \alpha \in \Sigma$, $d_1 \approx\!\!\!\mid\, \alpha$ if and only if $\exists d_2 \in$ SETOF(CHASE(CHASE(d, $\emptyset$, I), F, $\emptyset$)) such that $\forall \alpha \in \Sigma$, $d_2 \approx\!\!\!\mid\, \alpha$ and $d_1 = d_2$.

*Proof.* The if part is immediate, since SETOF(CHASE(CHASE(d, $\emptyset$, I), F, $\emptyset$)) $\subseteq$ SETOF(CHASE(d, F, I)). For the only if part assume that $d_1 \in$ SETOF(CHASE(d, F, I)) is such that $\forall \alpha \in \Sigma$, $d_1 \approx\!\!\!\mid\, \alpha$ and therefore $d_1$ is consistent. We need to show that $\exists d_2 \in$ SETOF(CHASE(CHASE(d, $\emptyset$, I), F, $\emptyset$)) such that $\forall \alpha \in \Sigma$, $d_2 \approx\!\!\!\mid\, \alpha$. The result is obtained by induction on the minimal number of chase steps required to arrive at $d_1$ during the computation of the chase. The basis step is vacuously true when zero chase steps are required to compute $d_1$. In the induction step there are two cases to consider.

Firstly, the last chase step is an application of the NFD rule. The result follows by inductive hypothesis which implies that the state of $d_1$ prior to the last chase step can be computed by applying the NIND rule first as long as possible.

Secondly, the last chase step is an application of the NIND rule for, say R[X] $\subseteq$ S[Y]. By inductive hypothesis the state of $d_1$ prior to the last chase step can be computed by applying the NIND rule first as long as possible. Thus we can assume that the penultimate chase step is an application of the NFD rule for T : W $\rightarrow$ Z.

Let $d'$ be the state of $d_1$ prior to the application of the last two chase steps associated with the NFD rule for T : W $\rightarrow$ Z and the NIND rule for R[X] $\subseteq$ S[Y] in that order. It can be verified that the equality with respect to the left-hand side of the NFD T : W $\rightarrow$ Z is not affected by the said NIND rule. Thus the NFD rule may be applied after the NIND rule has been applied to $d'$. Furthermore, by examining all the possibilities, the entries in the relations of $d_1$ attain their highest values with respect to $\sqsubseteq$ whether the aforesaid rules are interchanged or not. The result now follows by inductive hypothesis. $\square$

The next proposition is a consequence of Proposition 4.4 on using Propositions 4.1 and 4.3.

**Proposition 4.5** Let d be a consistent database over **R** and let $\Sigma = \text{F} \cup \text{I}$ be a set of NFDs and NINDs over **R**. Then $\exists d_1 \in$ POSS(d) such that $\forall \alpha \in \Sigma$, $d_1 \approx\!\!\!\mid\, \alpha$ if and only if $\exists d_2 \in$ SETOF(CHASE(d, F, I)) such that $\forall \alpha \in \Sigma$, $d_2 \approx\!\!\!\mid\, \alpha$. $\square$

The next proposition, whose proof is left to the reader, shows that the chase procedure is invariant with respect to sets of NFDs and NINDs that are equivalent with respect to logical implication.

**Proposition 4.6** Let d be a consistent database over **R** and let $\Sigma = \text{F} \cup \text{I}$ be a set of NFDs and NINDs over **R**. In addition, let F $\subseteq F'$ and I $\subseteq I'$ be sets of NFDs and NINDs over **R**, respectively, where $\forall \beta \in (F' \cup I') - \Sigma$, $\Sigma \approx\!\!\!\mid\, \beta$. Then $\exists d_1 \in$ SETOF(CHASE(d, F, I)) such that $\forall \alpha \in \Sigma$, $d_1 \approx\!\!\!\mid\, \alpha$ if and only if $\exists d_2 \in$ SETOF(CHASE(d, $F', I'$)) such that $\forall \alpha \in \Sigma$, $d_2 \approx\!\!\!\mid\, \alpha$. $\square$

From now on we will assume that when computing the chase we apply the NIND rule first as long as possible and then do not apply it any further, i. e. we compute CHASE(CHASE(d, $\emptyset$, I), F, $\emptyset$).

We close the section by defining a transformation, denoted by TR, which when applied to a database d over **R** returns a database TR(d, I) over **R** which satisfies all the NINDs in I by adding tuples to the relations in d. Specifically, TR(d, I) transforms d by applying the following transformation rule, initially to d, and then to the resulting states of d as long as possible, where $d'$ is a state of d during the transformation, $r'$ is the relation in $d'$ over R and $s'$ is the relation in $d'$ over S.

**Transformation rule:** if R[X] $\subseteq$ S[Y] $\in$ I and $\exists t_1 \in r'$ such that $\nexists t_2 \in s'$ with $t_1$[X] $\sqsubseteq t_2$[Y], then add a tuple $t_2$ over S to $s'$, with $t_2$[Y] $= t_1$[X] and $\forall$ A $\in$ schema(S) $-$ Y, $t_2$[A] $=$ *unk*.

It can be verified that $\forall \alpha \in$ I, TR(d, I) $\models_{\approx} \alpha$, by the definition of the satisfaction of a NIND, since otherwise the transformation rule would not have been applied as long as possible leading to a contradiction. We observe that the above transformation rule is utilised in Sections 7 and 8.

## 5 The additivity problem for null data dependencies

Herein we define the additivity problem referred to in the introduction.

**Definition 5.1 (Satisfaction of a set of NFDs and NINDs)** Let d be a database over **R** and $\Sigma$ be a set of NFDs and NINDs. Then d satisfies $\Sigma$, denoted by d $\models_{\approx} \Sigma$, whenever:

1. $\forall \alpha \in \Sigma$, d $\models_{\approx} \alpha$, and
2. $\exists d' \in$ POSS(d) such that $\forall \alpha \in \Sigma, d' \models_{\approx} \alpha$.

We note that by Proposition 3.1 if $\Sigma$ consists of NFDs only then in the above definition condition (1) is implied by condition (2).

Following [ATZE93] and [LEVE94a] we now define additive satisfaction of a class of sets of NFDs and NINDs with respect to a class of databases.

**Definition 5.2 (Additive satisfaction)** Satisfaction is said to be *additive* with respect to a class of databases, say *DBC*, and a class of sets of NFDs and NINDs, say *DEPC*, whenever:

$$\forall d \in DBC, \forall \Sigma \in DEPC, d \models_{\approx} \Sigma \text{ if and only if } \forall \alpha \in \Sigma, d \models_{\approx} \alpha.$$

Obviously, the only if part of the above definition trivially holds, since, by Definition 5.1, if $d \models_{\approx} \Sigma$ then $\forall \alpha \in \Sigma, d \models_{\approx} \alpha$. Thus in order to show that satisfaction is additive with respect to a class of databases and a class of sets of NFDs and NINDs we need to show that if $\forall \alpha \in \Sigma, d \models_{\approx} \alpha$, then $\exists d' \in$ POSS(d) such that $\forall \alpha \in \Sigma, d' \models_{\approx} \alpha$.

The next lemma follows immediately from Definitions 3.5 and 5.1.

**Lemma 5.1** Let d be a database over **R** and $\Sigma$ be a set of NFDs and NINDs over **R**. Then d $\models_{\approx} \Sigma$ if and only if d $\models_{\approx} \{\alpha \mid \Sigma \models_{\approx} \alpha\}$. $\square$

The following lemma is immediate, since condition (2) of Definition 5.1 vacuously holds for complete databases.

**Lemma 5.2** Satisfaction is additive with respect to COMPLETE(**R**) and NFD(**R**) $\cup$ NIND(**R**). $\square$

## 6 The additivity problem for null functional dependencies

Herein we investigate the additivity problem for NFDs on their own. The next lemma follows from the counterexamples provided in Example 1 of the introduction.

**Lemma 6.1** Satisfaction is not additive with respect to DB(**R**) and NFD(**R**). $\square$

We assume that the reader is familiar with Armstrong's axiom system for FDs [ARMS74, ULLM88, ATZE93]. The closure of a set of NFDs F over **R** with respect to Armstrong's axiom system, denoted by $F^+$, is defined by

$$F^+ = \bigcup \{\mathrm{F(R)}^+ \mid R \in \mathbf{R}\},$$

where $\mathrm{F(R)}^+$ is the closure of F(R) with respect to Arsmtrong's axiom system. A set of NFDs G over R is a *cover* of a set of NFDs F over R if $G^+ = F^+$.

**Definition 6.1 (Reduced and canonical sets of NFDs)** A NFD X $\rightarrow$ Y $\in$ $F^+$ is *reduced* [BEER79] if there does not exist a set of attributes W $\subset$ X such that W $\rightarrow$ Y $\in$ $F^+$. A set of NFDs F is *reduced* if all the NFDs in F are reduced.

A set of FDs F is *canonical* if it is reduced and the right-hand sides of all the FDs in F are singletons.

We further note that reduced and canonical covers G of a set of NFDs F can be obtained in polynomial time in the size of F [BEER79]. From now on for the rest of the paper we will restrict our attention to reduced sets of NFDs and assume that *all the sets of NFDs in* NFD(**R**) *are reduced.*

**Definition 6.2 (Monodependent sets of NFDs)** Two nontrivial NFDs of the forms R : X $\rightarrow$ A and R : Y $\rightarrow$ A, where A $\in$ schema(R), are said to be *incomparable* if X and Y are incomparable (i. e. X $\not\subseteq$ Y and Y $\not\subseteq$ X). Two nontrivial NFDs of the forms R : XB $\rightarrow$ A and R : YA $\rightarrow$ B, where A,B $\in$ schema(R), are said to be *cyclic.*

A set of NFDs F over **R** is *monodependent* if $\forall$ R $\in$ **R**, $\forall$ A $\in$ schema(R), the following two conditions are true:

1. Whenever there exist incomparable NFDs, X $\rightarrow$ A, Y $\rightarrow$ A $\in$ F(R)$^+$, then X $\cap$ Y $\rightarrow$ A $\in$ F(R)$^+$.
2. Whenever there exist cyclic NFDs, XB $\rightarrow$ A, YA $\rightarrow$ B $\in$ F(R)$^+$, then either Y $\rightarrow$ B $\in$ F(R)$^+$ or (X $\cap$ Y)A $\rightarrow$ B $\in$ F(R)$^+$.

An immediate consequence of the above definition is that if G is a cover of a set of NFDs F over **R**, then F is monodependent if and only if G is monodependent.

We observe that the two defining conditions of monodependent sets of NFDs correspond to the two defining properties of *conflict-free* sets of *multivalued dependencies* (MVDs) [SCIO81, LIEN82, BEER83]. In particular, condition (1) corresponds to the *intersection property* and condition (2) corresponds to the *split-freeness property.* We further observe that the set of MVDs that are logically implied by a monodependent set of NFDs may not be conflict-free and thus monodependence is a weaker notion than conflict-freeness. For example, let F(R) = {A $\rightarrow$ B, B $\rightarrow$ A} be a set of NFDs over R, with schema(R) = <A, B, C>. It can easily be verified that F(R) is monodependent but that the set of MVDs logically implied by F(R) is not conflict-free.

The next lemma shows that monodependence for NFDs implies additivity. In particular, the *pseudo-transitivity inference rule* will be utilised, where if R : X $\rightarrow$ Y $\in$ F(R)$^+$ and R : WY $\rightarrow$ Z $\in$ F(R)$^+$, then R : XW $\rightarrow$ Z $\in$ F(R)$^+$ can be derived.

**Lemma 6.2** Let F be a set of monodependent NFDs over **R**. Then $\forall d$ $\in$ DB(**R**), $\forall$ R $\in$ **R**, if $\forall$ R : X $\rightarrow$ Y $\in$ F, d $\models_{\approx}$ R: X $\rightarrow$ Y, then d $\models_{\approx}$ F.

*Proof.* Assume without loss of generality that F is a canonical set of NFDs noting that a canonical set of NFDs is reduced. Also assume that all the NFDs in F are of the form R : X $\rightarrow$ Y for a fixed R $\in$ **R**.

We show that if $\exists$ d $\in$ DB(**R**) such that $\forall$ R : X $\rightarrow$ Y $\in$ F, d $\models_{\approx}$ R: X $\rightarrow$ Y, but d $\not\models_{\approx}$ F, then F cannot be monodependent as assumed. Let d be such a database. By Proposition 4.1 it follows that CHASE(d, F, $\emptyset$) is inconsistent. We conclude the result by induction on the minimal number of chase steps, k, required to show that CHASE(d, F, $\emptyset$) is inconsistent.

(*Basis*): At least two chase steps are needed to show that CHASE(d, F, $\emptyset$) is inconsistent, since by assumption $\forall$ X $\rightarrow$ Y $\in$ F, d $\models_{\approx}$ X $\rightarrow$ Y. Thus consider the case when k = 2. Suppose the second chase step applies the NFD X $\rightarrow$ A $\in$ F to the tuples $t_1, t_2 \in$ r, where r $\in$ d is the relation over R. Then after the first chase step is applied $t_1$[X] = $t_2$[X] but $t_1$[A] $\sqcup$ $t_2$[A] is inconsistent.

Without loss of generality assume that the first chase step applies the NFD R: Y $\rightarrow$ B $\in$ F to the tuples $t_2$ and $t_3$ in r. There are two cases to consider.

*Case1:* B = A. In this case the first chase step applies the NFD Y $\rightarrow$ A to $t_2$ and $t_3$ in r resulting in $t_1$[A] $\sqcup$ $t_2$[A] being inconsistent. It follows that $t_1$[A] $\sqcup$ $t_3$[A] is inconsistent in r, since otherwise $t_1$[A] $\sqcup$ $t_2$[A] is inconsistent in r implying that one chase step is sufficient to show that CHASE(d, F, $\emptyset$) is inconsistent. Now, if X = Y then one chase step which applies the NFD X $\rightarrow$ A to the tuples $t_1$ and $t_3$ in r is sufficient to show that CHASE(d, F, $\emptyset$) is inconsistent, thus X $\neq$ Y. The result that F is not monodependent now follows by the assumption that F is canonical, since X $\rightarrow$ A and Y $\rightarrow$ A must be incomparable NFDs and X $\cap$ Y $\rightarrow$ A $\notin F^+$.

*Case2:* B $\neq$ A. In this case the first chase step applies the NFD Y $\rightarrow$ B to $t_2$ and $t_3$ in r resulting in $t_1$[B] = $t_2$[B]. Furthermore, $t_1$[A] $\sqcup$ $t_2$[A] is inconsistent in r, since k = 2. It follows that B $\in$ X, otherwise one chase step would suffice to show that CHASE(d, F, $\emptyset$) is inconsistent.

Now suppose that A $\notin$ Y. Then we can derive (X – B)Y $\rightarrow$ A $\in F^+$ by the pseudo-transitivity rule, where X $\not\subseteq$ (X – B)Y, since B $\notin$ Y. Now, if (X – B)Y $\not\subseteq$ X then (X – B)Y $\rightarrow$ A and X $\rightarrow$ A are incomparable. Furthermore, (X – B) is a proper subset of X and therefore (X – B) $\rightarrow$ A $\notin F^+$, since F is canonical. Thus, there exists a canonical set of NFDs G such that W $\rightarrow$ A, X $\rightarrow$ A $\in$ G, where W $\subseteq$ (X – B)Y. The result that F is not monodependent follows from Case1 by replacing F with G. Therefore, we assume that (X – B)Y $\subseteq$ X, implying that (X – B)Y is a proper subset of X, since B $\notin$ Y. This contradicts the fact that F is canonical, since (X – B)Y $\rightarrow$ A $\in F^+$.

We therefore suppose that A $\in$ Y. The result that F is not monodependent now follows, since Y $\rightarrow$ B and X $\rightarrow$ A are cyclic NFDs and in addition the following two assertions are true.

Firstly, (X – B) $\rightarrow$ A $\notin F^+$ due to the fact that by assumption F is canonical. Secondly, $t_1$[B] = $t_3$[B] in r, since Y $\rightarrow$ B was applied to $t_2$ and $t_3$ and k = 2. Moreover, $t_1$[(X $\cap$ Y)] = $t_3$[(X $\cap$ Y)] in r, since both $t_1$[(X $\cap$ Y)] = $t_2$[(X $\cap$ Y)] in r and $t_2$[(X $\cap$ Y)] = $t_3$[(X $\cap$ Y)] in r. Furthermore, $t_2$[A] = $t_3$[A] in r, since A $\in$ Y, and therefore $t_1$[A] $\sqcup$ $t_3$[A] is inconsistent in r. Thus, if (X $\cap$ Y)B $\rightarrow$ A $\in F^+$, then due to the assumption that F is canonical one chase step which applies the NFD (X $\cap$ Y)B $\rightarrow$ A to $t_1$ and $t_2$ would suffice to show that CHASE(d, F, $\emptyset$) is inconsistent. It therefore follows that (X $\cap$ Y)B $\rightarrow$ A $\notin F^+$ which is equivalent to ((X – B) $\cap$ (Y – A))B $\rightarrow$ A $\notin F^+$, since B $\notin$ Y and A $\notin$ X. (Recall that (X – B) $\rightarrow$ A $\notin F^+$.) The result that F is not monodependent follows.

(*Induction*): Assume the result holds when the minimal number of chase steps required to show that CHASE(d, F, $\emptyset$) is inconsistent is k, with k $\geq$ 2; we then need to prove that the result holds when the minimal number of chase steps required to show that CHASE(d, F, $\emptyset$) is inconsistent is k+1. Suppose the last chase step applies the NFD X $\rightarrow$ A $\in$ F to the tuples $t_1$ and $t_2$ in the penultimate state of r during the execution of CHASE(d, F, $\emptyset$). Then after the penultimate chase step is applied $t_1$[X] = $t_2$[X] but $t_1$[A] $\sqcup$ $t_2$[A] is inconsistent. Without loss of generality assume that the penultimate chase step applies the NFD Y $\rightarrow$ B $\in$ F to the tuples $t_2$ and $t_3$ in the state prior to the penultimate state of r during the execution of CHASE(d, F, $\emptyset$).

The result follows by a similar argument to the basis step noting that if k or less chase steps are sufficient to show that CHASE(d, F, $\emptyset$) is inconsistent, then the result follows by inductive hypothesis. □

The next lemma shows that additivity for NFDs implies monodependence.

**Lemma 6.3** Let F be a set of NFDs over **R** and assume that $\forall d \in$ DB(**R**), $\forall$ R $\in$ **R**, if $\forall$ R : X $\rightarrow$ Y $\in$ F, d $\approx\!\!\models$ R : X $\rightarrow$ Y, then d $\approx\!\!\models$ F. Then F is monodependent.

*Proof.* We show that if F is not monodependent, then $\exists$ d $\in$ DB(**R**) such that $\forall$ R : X $\rightarrow$ Y $\in$ F, d $\approx\!\!\models$ R : X $\rightarrow$ Y but d $\not\approx\!\!\models$ F. We assume without loss of generality that F(R) is not monodependent, where R $\in$ **R**, and $\forall$ S $\in$ **R** – {R}, F(S) is monodependent. Furthermore, we will assume for the rest of the proof that all NFDs are of the form R : X $\rightarrow$ Y. There are two cases to consider.

*Case1:* There exist incomparable NFDs X $\rightarrow$ A, Y $\rightarrow$ A $\in F^+$ but X $\cap$ Y $\rightarrow$ A $\notin F^+$. Let Rest = schema(R) $-$ XYA and let d $\in$ DB(**R**) be a database, where the relation r $\in$ d over R is shown in Table 11 and $\forall$ s $\in$ d $-$ {r}, s = $\emptyset$.

It can be verified that $\forall$ V $\rightarrow$ T $\in$ F, d $\models_{\approx}$ V $\rightarrow$ T, since the only NFDs that are not satisfied in d are of the form W $\rightarrow$ A, where W $\subseteq$ X $\cap$ Y, due to the assumption that X $\cap$ Y $\rightarrow$ A $\notin F^+$. The result now follows due to the fact that d $\not\models_{\approx}$ F, since it can easily be verified that $\forall d' \in$ POSS(d), either $d' \not\models_{\approx}$ X $\rightarrow$ A or $d' \not\models_{\approx}$ Y $\rightarrow$ A.

| X $\cap$ Y | X $-$ Y | Y $-$ X | A | Rest |
|---|---|---|---|---|
| $0 \ldots 0$ | $0 \ldots 0$ | $unk \ldots unk$ | 0 | $unk \ldots unk$ |
| $0 \ldots 0$ | $0 \ldots 0$ | $0 \ldots 0$ | $unk$ | $unk \ldots unk$ |
| $0 \ldots 0$ | $unk \ldots unk$ | $0 \ldots 0$ | 1 | $unk \ldots unk$ |

Table 11: The relation pertaining to Case1

*Case2:* There exist cyclic NFDs, XB $\rightarrow$ A, YA $\rightarrow$ B $\in F^+$, but both Y $\rightarrow$ B $\notin F^+$ and (X $\cap$ Y)A $\rightarrow$ B $\notin F^+$ obtain. Let Rest = schema(R) $-$ XYAB and let d $\in$ DB(**R**) be a database, where the relation r $\in$ d over R is shown in Table 12 and $\forall$ s $\in$ d $-$ {r}, s = $\emptyset$.

The only NFDs that are not satisfied in d are of the form W $\rightarrow$ B, where W $\subseteq$ Y, or of the form W $\rightarrow$ B, where W $\subseteq$ (X $\cap$ Y)A. These violations are justified due to the assumption that Y $\rightarrow$ B $\notin F^+$ and (X $\cap$ Y)A $\rightarrow$ B $\notin F^+$. The result now follows due to the fact that d $\not\models_{\approx}$ F, since it can easily be verified that $\forall d' \in$ POSS(r), $d' \not\models_{\approx}$ YA $\rightarrow$ B. □

| X $\cap$ Y | X $-$ Y | Y $-$ X | A | B | Rest |
|---|---|---|---|---|---|
| $0 \ldots 0$ | $0 \ldots 0$ | $unk \ldots unk$ | 0 | 0 | $unk \ldots unk$ |
| $0 \ldots 0$ | $0 \ldots 0$ | $0 \ldots 0$ | $unk$ | 0 | $unk \ldots unk$ |
| $0 \ldots 0$ | $unk \ldots unk$ | $0 \ldots 0$ | 0 | 1 | $unk \ldots unk$ |

Table 12: The relation pertaining to Case2

The following theorem summarises Lemmas 6.2 and 6.3.

**Theorem 6.4** Satisfaction is additive with respect to DB(**R**) and a class of sets NFDs **FC** if and only if all the sets of NFDs in **FC** are monodependent. □

**Theorem 6.5** Monodependence of a set of NFDs F over **R** can be checked in time polynomial in the size of F.

*Proof.* In [LEVE94a] it was shown that a canonical set of NFDs F(R) over R is monodependent if and only if $\forall$ A $\in$ schema(R), the following two conditions are true:

1. There exists at most one NFD in F(R) of the form X $\rightarrow$ A, and if X $\rightarrow$ A $\in$ F(R) then $\forall$ B $\in$ X, (schema(R) $-$ AB) $\rightarrow$ A $\notin$ F(R)$^+$.

2. Whenever X $\rightarrow$ A, Y $\rightarrow$ B $\in$ F(R), then either A $\notin$ Y or Y $\subseteq$ (X $\cap$ Y)A.

The result follows by [MAIE80], where it was shown that given a set of NFDs, G(R), a canonical set of NFDs, F(R), such that F(R)$^+$ = G(R)$^+$, can be obtained in time polynomial in the size of G(R). □

# 7 The additivity problem for unary null inclusion dependencies

Herein we investigate the additivity problem for unary NINDs on their own. The next lemma follows from the counterexample provided in Example 2 of the introduction.

**Lemma 7.1** Satisfaction is not additive with respect to DB(**R**) and NIND(**R**). $\square$

**Definition 7.1 (Monodirectional sets of unary NINDs)** A set of unary NINDs I over **R** is *monodirectional* if for every pair of nontrivial NINDs, R[A] $\subseteq$ S[B] and R[A] $\subseteq$ T[C], that are logically implied by I, at least one of the following conditions is true:

1. S = T and B = C, or
2. I $\models_{\approx}$ S[B] $\subseteq$ T[C], or
3. I $\models_{\approx}$ T[C] $\subseteq$ S[B], or
4. I $\models_{\approx}$ S[B] $\subseteq$ R[A], or
5. I $\models_{\approx}$ T[C] $\subseteq$ R[A].

The next lemma shows that monodirectionality implies additivity for unary NINDs.

**Lemma 7.2** Let I be a set of monodirectional unary NINDs over **R**. Then $\forall$ d $\in$ DB(**R**), if $\forall$ R[A] $\subseteq$ S[B] $\in$ I, d $\models_{\approx}$ R[A] $\subseteq$ S[B], then d $\models_{\approx}$ I.

*Proof.* We show that if $\exists$ d $\in$ DB(**R**) such that $\forall$ R[A] $\subseteq$ S[B] $\in$ I, d $\models_{\approx}$ R[A] $\subseteq$ S[B], but d $\not\models_{\approx}$ I, then I is not monodirectional. Let d be such a database. Then by Proposition 4.3, $\forall d' \in$ SETOF(CHASE(d, $\emptyset$, I)), $\exists$ R[A] $\subseteq$ S[B] $\in$ I such that $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B]. Let $d'$ be a database in SETOF(CHASE(d, $\emptyset$, I)) and R[A] $\subseteq$ S[B] $\in$ I be a unary NIND such that $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B].

We conclude the proof by induction on the minimal number, k, of chase steps required in the computation of $d'$ to derive the fact that $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B].

(*Basis*): Assume that k = 1 and thus the chase step applies the NIND rule for R[A] $\subseteq$ T[C] $\in$ I to d. The application of this NIND rule must cause a distinguished nonnull value, say $\perp_i$, in Dum to be modified to a nonnull value, say $v$, in *Dom* $-$ Dum. Furthermore, $d' \models_{\approx}$ R[A] $\subseteq$ T[C], as a result of applying the NIND rule for R[A] $\subseteq$ T[C]. We now show that all the conditions of monodirectionality given in Definition 7.1 are violated, otherwise a contradiction arises.

1. If S = T and B = C then it follows that $d' \models_{\approx}$ R[A] $\subseteq$ S[B].
2. Assume that I $\models_{\approx}$ S[B] $\subseteq$ T[C] and thus d $\models_{\approx}$ S[B] $\subseteq$ T[C] by the definition of logical implication. By modifying $\perp_i$, instead of to $v$, to an appropriate nonnull value in the active domain of the relation over S in d we obtain $d' \models_{\approx}$ R[A] $\subseteq$ S[B], since d $\models_{\approx}$ R[A] $\subseteq$ S[B].
3. Assume that I $\models_{\approx}$ T[C] $\subseteq$ S[B] and thus d $\models_{\approx}$ T[C] $\subseteq$ S[B] by the definition of logical implication. It follows that $d' \models_{\approx}$ R[A] $\subseteq$ S[B], since d $\models_{\approx}$ R[A] $\subseteq$ S[B].
4. Assume that I $\models_{\approx}$ S[B] $\subseteq$ R[A] and thus d $\models_{\approx}$ S[B] $\subseteq$ R[A] by the definition of logical implication. Therefore, I $\models_{\approx}$ S[B] $\subseteq$ T[C] by the transitivity rule for NINDs [CASA84, LEVE94b] and thus d $\models_{\approx}$ S[B] $\subseteq$ T[C]. The result follows as in (2).
5. Assume that I $\models_{\approx}$ T[C] $\subseteq$ R[A] and thus d $\models_{\approx}$ T[C] $\subseteq$ R[A] by the definition of logical implication. Therefore, I $\models_{\approx}$ T[C] $\subseteq$ S[B] by the transitivity rule for NINDs and thus d $\models_{\approx}$ T[C] $\subseteq$ S[B]. The result follows as in (3).

(*Induction*): Assume the result holds when the minimal number of chase steps required to to compute $d'$ is k, with k $\geq$ 1; we then need to prove that the result holds when the minimal number of chase steps required to derive the fact that $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B] is k+1. Let $d_1$ be the state of $d'$ prior to applying the (k+1)th chase step and let the (k+1)th chase step be an application of the NIND rule for a NIND Q[D] $\subseteq$ T[C] $\in$ I. By inductive hypothesis $d_1 \models_{\approx}$ R[A] $\subseteq$ S[B]; moreover, $d' \models_{\approx}$ Q[D] $\subseteq$ T[C]. There are two cases to consider.

*Case1:* I $\models_{\approx}$ R[A] $\subseteq$ Q[D] obtains. By inductive hypothesis $d_1 \models_{\approx}$ R[A] $\subseteq$ Q[D], since otherwise we can replace R[A] $\subseteq$ S[B] by R[A] $\subseteq$ Q[D] and obtain the result in k chase steps. By the transitivity rule for NINDs it follows that I $\models_{\approx}$ R[A] $\subseteq$ T[C] and thus $d' \models_{\approx}$ R[A] $\subseteq$ T[C]. The result is now obtained by an argument analogous to that made in the basis step utilising Proposition 4.6.

*Case2:* I $\not\models_{\approx}$ R[A] $\subseteq$ Q[D] obtains. There are two subcases to consider.

Firstly, I $\models_{\approx}$ Q[D] $\subseteq$ R[A] obtains. By inductive hypothesis $d_1 \models_{\approx}$ Q[D] $\subseteq$ R[A], since otherwise we can replace R[A] $\subseteq$ S[B] by Q[D] $\subseteq$ R[A] and obtain the result in k chase steps. By the transitivity rule for NINDs it follows that I $\models_{\approx}$ Q[D] $\subseteq$ S[B] and thus $d_1 \models_{\approx}$ Q[D] $\subseteq$ S[B]. Next, suppose that the (k+1)th chase step causes a distinguished nonnull value, say $\perp_i$, in $d_1$ to be modified to a nonnull value, say $v$, in $Dom$ − Dum. It follows that $\perp_i$ is an A-value in the relation over R in $d_1$ and that $v$ is not a B-value in the relation over S in $d'$, since $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B]. Therefore, $d' \not\models_{\approx}$ Q[D] $\subseteq$ S[B], since $v$ is a D-value in the relation over Q in $d'$. The result follows by replacing R[A] $\subseteq$ S[B] by Q[D] $\subseteq$ S[B] and arguing that I is not monodirectional in a manner analogous to that made in the basis step on utilising Proposition 4.6.

Secondly, I $\not\models_{\approx}$ Q[D] $\subseteq$ R[A] obtains. There must therefore exist a distinguished nonnull value, say $\perp_i$, in Dum that is both an A-value of the relation over R in $d_1$ and a D-value of the relation over Q in $d_1$, since $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B]. There are two final subcases to consider.

Firstly, for some relation schema P and an attribute E $\in$ schema(P), both I $\models_{\approx}$ P[E] $\subseteq$ R[A] and I $\models_{\approx}$ P[E] $\subseteq$ Q[D] obtain. The result that I is not monodirectional follows by an argument analogous to that made in the basis step, where R[A] is replaced by P[E], on utilising Proposition 4.6.

Secondly, for some relation schema P and an attribute E $\in$ schema(P), both I $\models_{\approx}$ R[A] $\subseteq$ P[E] and I $\models_{\approx}$ Q[D] $\subseteq$ P[E] obtain. By inductive hypothesis, $d_1 \models_{\approx}$ R[A] $\subseteq$ P[E], since otherwise we can replace R[A] $\subseteq$ S[B] by R[A] $\subseteq$ P[E] and obtain the result in k or less chase steps. Furthermore, P[E] $\neq$ S[B] and $d' \models_{\approx}$ Q[D] $\subseteq$ P[E], since otherwise the result that I is not monodirectional follows by an argument analogous to that made in the basis step, where R[A] is replaced by Q[D] and S[B] is replaced by P[E], on utilising Proposition 4.6. We can therefore assume without loss of generality that P[E] = T[C], since an application of the NIND Q[D] $\subseteq$ P[E] at the appropriate stage during the computation of $d'$ will cause the violation $d' \not\models_{\approx}$ R[A] $\subseteq$ S[B]. It follows that $d' \models_{\approx}$ R[A] $\subseteq$ P[E], since the application of Q[D] $\subseteq$ P[E] replaces $\perp_i$ by an E-value in the relation over P in $d_1$. The result that I is not monodirectional follows by an argument analogous to that made in the basis step, where R[A] $\subseteq$ T[C] is replaced by R[A] $\subseteq$ P[E], on utilising Proposition 4.6. □

The next lemma shows that additivity for unary NINDs implies monodirectionality.

**Lemma 7.3** Let I be a set of unary NINDs over **R** and assume that $\forall$ d $\in$ DB(**R**), if $\forall$ R[A] $\subseteq$ S[B] $\in$ I, d $\models_{\approx}$ R[A] $\subseteq$ S[B], then d $\models_{\approx}$ I. Then I is monodirectional.

*Proof.* We show that if I is not monodirectional, then $\exists$ d $\in$ DB(**R**) such that $\forall$ R[A] $\subseteq$ S[B] $\in$ I, d $\models_{\approx}$ R[A] $\subseteq$ S[B] but d $\not\models_{\approx}$ I. Now, if I is not monodirectional then there exists a pair of nontrivial unary NINDs R[A] $\subseteq$ S[B] and R[A] $\subseteq$ T[C] that are logically implied by I, such that either S $\neq$ T or B $\neq$ C and *none* of the following conditions is true:

1. I $\models_{\approx}$ S[B] $\subseteq$ T[C].

2. I $\models_{\approx}$ T[C] $\subseteq$ S[B].

3. I $\models_{\approx}$ S[B] $\subseteq$ R[A].

4. I $\models_{\approx}$ T[C] $\subseteq$ R[A].

Let d be a database over **R**, where r $\in$ d is the relation over R shown in Table 13, s $\in$ d is the relation over S shown in Table 14, z $\in$ d is the relation over T shown in Table 15, and all the other relations in d $-$ {r, s, z} are empty.

| schema(R) |
|---|
| $unk \ldots unk$ |

Table 13: The counterexample relation r over R

| B | schema(S)$-$B |
|---|---|
| 0 | $unk \ldots unk$ |

Table 14: The counterexample relation s over S

| C | schema(T)$-$C |
|---|---|
| 1 | $unk \ldots unk$ |

Table 15: The counterexample relation z over T

Let $r_1$ in TR(d, I) be the relation over R, $s_1$ in TR(d, I) be the relation over S and $z_1$ in TR(d, I) be the relation over T, where TR is the transformation defined at the end of Section 4.

We claim that *none* of the following statements is true:

1. $0 \in \pi_C(z_1)$.
2. $1 \in \pi_B(s_1)$.
3. $0 \in \pi_A(r_1)$.
4. $1 \in \pi_A(r_1)$.

The claim follows, since if one or more of the above statements is true, then I would be monodirectional. More specifically, by a straightforward induction on the number of times the transformation rule was applied to d it can be shown that:

1. if (1) is true then I $\models_{\approx}$ S[B] $\subseteq$ T[C],
2. if (2) is true then I $\models_{\approx}$ T[C] $\subseteq$ S[B],
3. if (3) is true then I $\models_{\approx}$ S[B] $\subseteq$ R[A] and
4. if (4) is true then I $\models_{\approx}$ T[C] $\subseteq$ R[A].

The result now follows, since TR(d, I) $\not\models_{\approx}$ I is evident. □

The following theorem summarises Lemmas 7.2 and 7.3.

**Theorem 7.4** Satisfaction is additive with respect to DB(**R**) and a class of sets of unary NINDs **IC** if and only if all the sets of NINDs in **IC** are monodirectional. □

The following theorem is an immediate consequence of Definition 7.1 and Corollary 3.4.

**Theorem 7.5** Monodirectionality of a set I of unary NINDs over **R** can be checked in time polynomial in the size of I. □

We conclude this section by showing that when the NINDs are not unary then we cannot just replace A by X, B by Y and C by W in Definition 7.1. Let r, s and z be the relations shown in Tables 16, 17 and 18, respectively, and let d = {r, s, z} be a database over **R**, with **R** = {R, S, T}, schema(R) = <A, B>, schema(S) = <C, D> and schema(T) = <E>. Furthermore, let I = {R[AB] $\subseteq$ S[CD], R[B] $\subseteq$ T[E], T[E] $\subseteq$ S[C], T[E] $\subseteq$ S[D]}. It can easily be verified that $\forall \alpha \in$ I, d $\models_{\approx} \alpha$ but d $\not\models_{\approx}$ I. Moreover, the set of unary NINDs implied by I is monodirectional.

| A | B |
|---|---|
| 1 | *unk* |

Table 16: The counterexample relation r

| C | D |
|---|---|
| 0 | 1 |
| 1 | 0 |

Table 17: The counterexample relation s

| E |
|---|
| 1 |

Table 18: The counterexample relation z

## 8 The mixed additivity problem

Herein we investigate the additivity problem for NFDs and unary NINDs. The next lemma follows from the counterexample provided in Example 3 of the introduction.

**Lemma 8.1** Satisfaction is not additive with respect to DB($\mathbf{R}$) and the class of sets of NFDs and NINDs that contains only a single NFD and a single NIND. $\square$

We now solve the mixed additivity problem for a set $\Sigma$ = F $\cup$ I, where I is a set of unary NINDs. Without loss of generality we assume that $\Sigma$ does not contain any trivial NFDs or NINDs and that the right-hand sides of NFDs in F are singletons. Moreover, we will assume that all the NFDs and NINDs mentioned from now on are nontrivial.

**Definition 8.1 (The cyclicity condition)** $\Sigma$ satisfies the *cyclicity condition* if whenever $\Sigma \models_{\approx}$ R : XA $\rightarrow$ B, $\Sigma \not\models_{\approx}$ R : X $\rightarrow$ A, $\Sigma \not\models_{\approx}$ R : X $\rightarrow$ B and $\Sigma \models_{\approx}$ R[A] $\subseteq$ S[C], then $\Sigma \models_{\approx}$ R : B $\rightarrow$ A and $\Sigma \models_{\approx}$ R[B] $\subseteq$ S[C].

We observe that if X = $\emptyset$ then, since we have assumed that NFDs are standard, the if part of Definition 8.1 can be simplified to: whenever $\Sigma \models_{\approx}$ R : A $\rightarrow$ B and $\Sigma \models_{\approx}$ R[A] $\subseteq$ S[C]. Furthermore, if F is monodependent and $\Sigma \models_{\approx}$ R : XA $\rightarrow$ B, $\Sigma \not\models_{\approx}$ X $\rightarrow$ B and $\Sigma \models_{\approx}$ R : B $\rightarrow$ A, then it must be the case that $\Sigma \models_{\approx}$ R : A $\rightarrow$ B.

The next lemma shows that monodependence and monodirectionality together with the cyclicity condition imply additivity for NFDs and unary NINDs.

**Lemma 8.2** Let $\Sigma$ = F $\cup$ I be a set of NFDs and unary NINDs over $\mathbf{R}$, where F is monodependent, I is monodirectional and $\Sigma$ satisfies the cyclicity condition. Then $\forall$ d $\in$ DB($\mathbf{R}$), if $\forall \alpha \in \Sigma$, d $\models_{\approx} \alpha$, then d $\models_{\approx} \Sigma$.

*Proof.* Assume, without loss of generality, that F is a canonical set of NFDs noting that a canonical set of NFDs is reduced. We show that if $\exists$ d $\in$ DB($\mathbf{R}$) such that $\forall \alpha \in \Sigma$, d $\models_{\approx} \alpha$ but d $\not\models_{\approx} \Sigma$, then either F is not monodependent, or I is not monodirectional or $\Sigma$ violates the cyclicity condition. Let d be such a database. If I is not monodirectional then the result follows by Theorem 7.4 on using Proposition 4.4. If F is not monodependent (assuming that I is monodirectional) then the result follows by Theorem 6.4, since there is no interaction between F and I by Theorem 3.3 due to the fact that I is a set of unary NFDs. (See the example at the end of this section which shows how monodependence needs to be modified when the NINDs may not be unary.)

So assume that F is monodependent and that I is monodirectional. Thus by Proposition 4.5, $\forall d' \in$ SETOF(CHASE(d, F, I)), $\exists \beta \in \Sigma$ such that $d' \not\models_{\approx} \beta$. Moreover, by Proposition 4.4 $\forall d' \in$ SETOF(CHASE(CHASE(d, $\emptyset$, I), F, $\emptyset$)), $\exists \beta \in \Sigma$ such that $d' \not\models_{\approx} \beta$. Let $d_2 \in$ SETOF(CHASE(CHASE(d, $\emptyset$, I), F, $\emptyset$)) be such a database, with $d_1$ = CHASE(d, $\emptyset$, I), and thus $d_2$ = CHASE($d_1$, F, $\emptyset$). Now, since I is monodirectional it must be the case that $\forall \alpha \in$ I, $d_1 \models_{\approx} \alpha$. Furthermore, there exists a NFD or NIND $\beta \in \Sigma$ such that $d_2 \not\models_{\approx} \beta$. It follows that $d_2$ is inconsistent, i. e. that $\beta$ is a NFD, due to the fact that I is a set of unary NINDs which implies that no application of the NFD rule for a NFD in F can lead to a NIND in I being violated. Moreover, $\exists \beta \in$ F such that

$d_1 \not\approx \beta$, since otherwise $d_2$ must be consistent due to the fact that F is monodependent, leading to a contradiction. Thus only one chase step for a NFD in F need be applied to $d_1$ during the computation of CHASE($d_1$, F, $\emptyset$) in order to show that $d_2$ is inconsistent.

Assume that the NFD in F leading to $d_2$ being inconsistent by applying one chase step to $d_1$ is R : XA $\rightarrow$ B. Therefore, $\Sigma \not\approx$ R : X $\rightarrow$ B and $\Sigma \not\approx$ R : X $\rightarrow$ A, due to our assumption that F is canonical. Furthermore, assume that R : XA $\rightarrow$ B is applied to the tuples $t_1, t_2 \in$ r, with $t_1$[XA] = $t_2$[XA] and $t_1$[B] $\sqcup$ $t_2$[B] being inconsistent, where r $\in d_1$ is the relation over R. Without loss of generality we assume that the inconsistency arises due to a modification of a prior state $u_1$ of $t_1$ by an application of the NIND rule for a NIND in I. There are two cases to consider.

*Case1:* In the first case an application of the NIND rule causes $u_1$[B] to be modified to $t_1$[B], where $u_1[\mathrm{B}] = \perp_i \in$ Dum and $t_1[\mathrm{B}] = 1 \in$ *Dom* $-$ Dum are two B-values of the relation r over R in the state of $d_1$ prior to the application of the NIND rule. We show that this case cannot occur by contradiction to the fact that $d_1 \not\approx$ R : XA $\rightarrow$ B.

There are two subcases to consider, where, without loss of generality, $u_1$[AB] and $t_2$[AB] are shown in Table 19 and a C-value, in the active domain of a relation s over S in the state of $d_1$ prior to the application of the NIND rule, is shown in Table 20.

Firstly, the NIND rule for S[C] $\subseteq$ R[B] is applied, assuming that this NIND is in I. But then the nonnull value 1 is the B-value in the active domain of r, so $d_1 \approx$ S[C] $\subseteq$ R[B]. Thus the NIND rule for S[C] $\subseteq$ R[B] need not be applied, leading to a contradiction.

Secondly, the NIND rule for R[B] $\subseteq$ S[C] is applied, assuming that this NIND is in I. But then the nonnull value 0 must be a C-value in the active domain of s, since $d_1 \approx$ R[B] $\subseteq$ S[C]. Thus we can modify $\perp_i$ to 0, also leading to a contradiction of the fact that $d_1 \not\approx$ R : XA $\rightarrow$ B.

| A | B |
|---|---|
| 0 | $\perp_i$ |
| 0 | 0 |

Table 19: The counterexample tuples in r

| C |
|---|
| 1 |

Table 20: The counterexample tuple in s

*Case2:* In the second case an application of the NIND rule for R[A] $\subseteq$ S[C], assuming that this NIND is in I, causes $u_1$[A] to be modified to $t_1$[A], where $u_1[\mathrm{A}] = \perp_i \in$ Dum and $t_1[\mathrm{A}] = 0 \in$ *Dom* $-$ Dum are two A-values of the relation r over R in the state of $d_1$ prior to the application of the NIND rule. We show by contradiction that this case cannot occur unless the cyclicity condition is violated. Again, there are two subcases to consider.

Firstly, without loss of generality, $u_1$[AB] and $t_2$[AB] are shown in Table 21 and a C-value, in the active domain of a relation s over S in $d_1$ prior to the application of the NIND rule, is shown in Table 22. We assume in this case that if $\perp_i$ can be modified to 1, then the contradiction pertaining to $d_1 \not\approx$ R : XA $\rightarrow$ B will not arise. Now, by the cyclicity condition, $\Sigma \approx$ R[B] $\subseteq$ S[C] and thus $d_1 \approx$ R[B] $\subseteq$ S[C] by the definition of logical implication due to the fact that $\forall \alpha \in$ I, $d_1 \approx \alpha$. Therefore, 1 must also be a C-value in the active domain of s and thus we can modify $\perp_i$ to 1 because of the NIND rule associated with R[A] $\subseteq$ S[C], leading to a contradiction of the fact that $d_1 \not\approx$ R : XA $\rightarrow$ B.

| A | B |
|---|---|
| $\perp_i$ | 0 |
| 0 | 1 |

Table 21: The counterexample tuples in r

| C |
|---|
| 0 |

Table 22: The counterexample tuple in s

Secondly, without loss of generality, $u_1$[AB], $t_2$[AB] and a third AB-value $t_3$[AB] are shown in Table 23, and two C-values, in the active domain of a relation s over S in $d_1$ prior to the application of the NIND rule, are shown in Table 24. Furthermore, assume that the state of $d_1$, prior to the application of the NIND rule for R[A] $\subseteq$ S[C], satisfies R[B] $\subseteq$ S[C] and that $\perp_i$ must be modified either to 0 or to 1. It can be verified that this state of $d_1$ violates R : B $\rightarrow$ A. Now, d $\models\approx$ R : B $\rightarrow$ A by the definition of logical implication due to the fact that $\forall \alpha \in$ F, d $\models\approx \alpha$ and furthermore by the cyclicity condition $\Sigma \models\approx$ R : B $\rightarrow$ A. Therefore, in a manner similar to that concerning the violation of R : XA $\rightarrow$ B in $d_1$, a NIND rule in I must have caused $d_1$ to violate R : B $\rightarrow$ A. The result now follows, since if $t_2$[A] = 0 then $u_1$[A] can be modified to 1 and if $t_3$[A] = 1 then $u_1$[A] can be modified to 0, both modifications leading to a contradiction of the fact that $d_1 \not\models\approx$ R : XA $\rightarrow$ B. (The modifications are due to the NIND rule for R[A] $\subseteq$ S[C].) $\square$

| A | B |
|---|---|
| $\perp_i$ | 0 |
| 0 | 1 |
| 1 | 1 |

Table 23: The counterexample tuples in r

| C |
|---|
| 0 |
| 1 |

Table 24: The counterexample tuples in s

The next lemma shows that additivity for NFDs and unary NINDs implies monodependence, monodirectionality and the cyclicity condition.

**Lemma 8.3** Let $\Sigma$ be a set of NFDs and unary NINDs over **R** and assume that $\forall$ d $\in$ DB(**R**), if $\forall \alpha \in \Sigma$, d $\models\approx \alpha$, then d $\models\approx \Sigma$. Then F is monodependent, I is monodirectional and $\Sigma$ satisfies the cyclicity condition.

*Proof.* We show that if F is not monodependent, or if I is not monodirectional, or if the cyclicity condition is violated, then $\exists$ d $\in$ DB(**R**) such that $\forall \alpha \in \Sigma$ d $\models\approx \alpha$ but d $\not\models\approx \Sigma$. In order to prove the result we consider the three cases in turn.

*Case1.* Suppose that F is not monodependent. We define an auxiliary relation, designated R(0, 1), over R $\in$ **R**. Let R(i), where i $\in$ {0,1}, be defined by R(i) = $\cup_{A \in schema(R)}$ $\{t_A \mid \forall$ B $\in$ schema(R)$-$A, $t_A$[B] = *unk* and $t_A$[A] = i}. We define R(0, 1) by R(0, 1) = R(0) $\cup$ R(1). The result now follows by unioning R(0, 1) to all the relations constructed in Lemma 6.3, noting that r satisfies all the NFDs in F(R) if and only if r $\cup$ R(0, 1) also satisfies all the NFDs in F(R).

*Case2.* Suppose that I is not monodirectional. The result follows by Lemma 7.3 noting that TR(d, I) constructed in the proof of that lemma also satisfies all the NFDs in F.

*Case3.* Suppose that $\Sigma$ does not satisfy the cyclicity condition. Then there exists a NFD R : XA $\rightarrow$ B and a NIND R[A] $\subseteq$ S[C] such that $\Sigma \models\approx$ R : XA $\rightarrow$ B, $\Sigma \not\models\approx$ R : X $\rightarrow$ A, $\Sigma \not\models\approx$ R : X $\rightarrow$ B and $\Sigma \models\approx$ R[A] $\subseteq$ S[C], but either $\Sigma \not\models\approx$ R : B $\rightarrow$ A or $\Sigma \not\models\approx$ R[B] $\subseteq$ S[C].

Firstly, suppose that $\Sigma \not\models\approx$ R : B $\rightarrow$ A. Let d be a database over **R**, with r $\in$ d being the relation over R shown in Table 25, s $\in$ d being the relation over S shown in Table 26, and all the other relations in d $-$ {r, s} being empty. It can be verified that $\forall \alpha \in \Sigma$, TR(d, I) $\models\approx \alpha$, where TR is the transformation defined at the end of Section 4. It is also evident that TR(d, I) $\not\models\approx \Sigma$ as required.

Secondly, suppose that $\Sigma \not\models\approx$ R[B] $\subseteq$ S[C]. Let d be a database over **R**, with r $\in$ d being the relation over R shown in Table 27, s $\in$ d being the relation over S shown in Table 28, and all the other relations in d $-$ {r, s} being empty. It can be verified that $\forall \alpha \in \Sigma$, TR(d, I) $\models\approx \alpha$. It is also evident that TR(d, I) $\not\models\approx \Sigma$ as required. $\square$

The following theorem summarises Lemmas 8.2 and 8.3.

| X | A | B | schema(R)−XAB |
|---|---|---|---|
| $0 \ldots 0$ | 0 | 0 | $unk \ldots unk$ |
| $0 \ldots 0$ | 1 | 0 | $unk \ldots unk$ |
| $0 \ldots 0$ | $unk$ | 1 | $unk \ldots unk$ |

Table 25: The first counterexample relation r over R

| schema(S) |
|---|
| $0 \ldots 0$ |
| $1 \ldots 1$ |

Table 26: The first counterexample relation s over S

| X | A | B | schema(R)−XAB |
|---|---|---|---|
| $0 \ldots 0$ | 0 | 0 | $unk \ldots unk$ |
| $0 \ldots 0$ | $unk$ | 1 | $unk \ldots unk$ |

Table 27: The second counterexample relation r over R

| schema(S) |
|---|
| $0 \ldots 0$ |

Table 28: The second counterexample relation s over S

**Theorem 8.4** Satisfaction is additive with respect to DB(**R**) and a class of sets NFDs and unary NINDs **FC** ∪ **IC**, where **FC** and **IC** are classes of sets NFDs and unary NINDs, respectively, if and only if all the sets of NFDs in **FC** are monodependent, all the sets of NINDs in **IC** are monodirectional and all the sets of NFDs and NINDs in **FC** ∪ **IC** satisfy the cyclicity condition. □

The next theorem is an immediate consequence of Definition 8.1, Corollary 3.4, Theorem 6.5 and Theorem 7.5.

**Theorem 8.5** Whether or not a set $\Sigma$ of NFDs and unary NINDs satisfies the cyclicity condition can be checked in time polynomial in the size of $\Sigma$. □

We conclude this section by demonstrating that when the NINDs are not unary the solution to the mixed additivity problem requires the extension of the definition of monodependence. Let F = {S : D → E} and I = {R[AC] ⊆ S[DE], R[BC] ⊆ S[DE]}, with schema(R) = <A, B, C> and schema(S) = <D, E>. The NFDs R : A → C and R : B → C can be derived from F ∪ I by the *pullback inference rule* [MITC83, LEVE94b]. In addition, F is not monodependent, since F ∪ I $\not\approx$ $\emptyset \rightarrow$ C (in any case we have assumed that all NFDs are standard). Next, let d = {r, s} be a database where r over R is shown in Table 29 and s over S is shown in Table 30; we assume that $\perp_1$ and $\perp_2$ are distinguished nonnull values in *Dom*. It can easily be verified that d $\models\approx$ R : B → C, d $\models\approx$ R : A → C, d $\models\approx$ R[BC] ⊆ S[DE] and d $\models\approx$ R[AC] ⊆ S[DE]. Moreover, d $\models\approx$ S : D → E if and only if $\perp_1 = \perp_2$ and thus d $\models\approx$ F ∪ I if and only if $\perp_1 = \perp_2$. Therefore, if d $\models\approx$ S : D → E then d $\models\approx$ F ∪ I. We have thus exhibited a set F ∪ I of NFDs and NINDs in which F is not monodependent but for which satisfaction is additive.

| A | B | C |
|---|---|---|
| 0 | $unk$ | $\perp_1$ |
| 0 | 0 | $unk$ |
| $unk$ | 0 | $\perp_2$ |

Table 29: The counterexample relation r over R

| D | E |
|---|---|
| 0 | $\perp_1$ |
| 0 | $\perp_2$ |

Table 30: The counterexample relation s over S

## 9 A Brief Survey of Related Work

Herein we review the history of FDs in the context of incomplete relational databases as it relates to the results presented in this paper.

Grant [GRAN80] was one of the pioneers in considering the effect of null values on the satisfaction of FDs. Translating Grant's definition of a *maybe* FD into our terminology, X $\rightarrow$ Y is satisfied in a relation r over R, if whenever $t_1[X] \sqcup t_2[X]$ is consistent, then $t_1[Y] \sqcup t_2[Y]$ is also consistent, where $t_1, t_2 \in$ r. Grant did not consider either the axiomatisation of NFDs or the satisfaction of a set of NFDs in a relation.

Vassiliou [VASS80] and Honeyman [HONE82] considered the satisfaction of a set of NFDs in a relation, their main results being Propositions 4.1 and 4.2. Vassiliou also considered the axiomatisation of NFDs but only in the restricted case when the FDs are *strong*, namely in the sense that they are required to be satisfied in all possible worlds. It is interesting to note that the context of Honeyman's work is not incomplete information but rather that of determining the satisfaction of a set of FDs defined over a relation schema R, with respect to a database d over $\mathbf{R}$, and where schema(R) $= \bigcup_{i=1}^{n}$ schema$(R_i)$.

Lien [LIEN82] as well as Atzeni and Morfuni [ATZE84] considered the axiomatisation of NFDs. In particular, they showed that the inference rules: reflexivity, augmentation, decomposition and union, are sound and complete for NFDs; we call this axiom system, Lien and Atzeni's axiom system. That is, by dropping the transitivity rule from Armstrong's axiom system and adding the decomposition and union rules, we obtain Lien and Atzeni's axiom system. In [LEVE94a] we have shown that when a set of NFDs is reduced and monodependent then the closure of such a set of NFDs, with respect to Lien and Atzeni's axiom system, is the same as its closure with respect to Armstrong's axiom system. In contrast to our work, Atzeni and Morfuni considered the semantics of unmarked nulls to be "no information", and Lien considered the semantics of unmarked nulls to be "does not exist" or "inapplicable". Lien and Atzeni's sound and complete axiom system was preempted by [LIEN79], wherein the axiomatisation of multivalued dependencies in incomplete relations is considered. In [LEVE95a] we have shown that Lien and Atzeni's axiom system can be viewed in terms propositional implications holding in a subset of a three-valued propositional logic. In [ATZE86] the interaction of *existence constraints* with NFDs is also considered.

As far as we are aware the first explicit and unambiguous formulation of the additivity problem was given in [ATZE93, page 243], but a solution to this problem was not forthcoming with respect to the implication problem. We consider the solution of the additivity problem to be an important prerequisite for any relational database system supporting NFDs, since otherwise semantic anomalies may arise. The results obtained herein make a contribution in this direction.

A formalisation of functional and multivalued dependencies in incomplete relations from a domain theoretic point of view was carried out by Libkin [LIBK91], thus extending the work of [BUNE91]. The motivation behind Libkin's research is to establish the semantics of types in database programming languages. Additionally, Levene and Loizou [LEVE92] investigate join dependencies in the context of incomplete relations. The main result in [LEVE92] is that a variant of the chase procedure, called the *or-chase*, is a sound and complete inference procedure for null join dependencies. Finally, Levene and Loizou [LEVE94b] introduced the NIND and exhibited a sound and complete axiom system for NFDs and NINDs. Moreover, Levene and Loizou showed that the implication problem for NFDs and NINDs is decidable and EXPTIME-complete (for FDs and INDs it was shown that the implication problem is undecidable [MITC83]).

An extension of the concept of keys, which relaxes entity integrity in the context of incomplete information, was investigated by Thalheim [THAL89]. A family $\mathbf{K}$ of subsets of schema(R) is called a *key set* for a relation r over R, if $\forall t_1, t_2 \in$ r, $\exists K \in \mathbf{K}$ such that $t_1[K]$ and $t_2[K]$ are complete and $t_1[K] = t_2[K]$.

Imielinski and Lipski [IMIE83] consider the more general framework of tuple and equality generating dependencies and Grahne [GRAH91] extends their work building on [IMIE84]. Both Imielinski and Lipski, and Grahne deal mainly with *marked* nulls as opposed to the unmarked nulls considered herein, and are interested in the problem of finding the most suitable representation for incomplete databases. The additivity problem does not arise in their work as such.

Our definition of NFD embodies the *Closed World Assumption* (CWA) [IMIE84, LIBK94], since our definition of the set of possible worlds of a database, d, (Definition 2.9) allows only completed tuples from the relations in d to be present in POSS(d). On the other hand, if we had taken the *Open World Assumption* (OWA) [IMIE84, LIBK94], Definition 2.9 would have to be relaxed to allow additional tuples to be present in POSS(d), which do not result from completed tuples in the relations of d. It can be shown that the results obtained herein also hold when the OWA approach is taken, by considering the following argument. Suppose that d $\models\!\approx \Sigma$, where d is a database over $\mathbf{R}$ and $\Sigma$ is a set of NFDs and NINDs over $\mathbf{R}$, and let $d_1 \in$ POSS(d) be a database such that $\forall \alpha \in \Sigma, d_1 \models\!\approx \alpha$; the existence of $d_1$ is guaranteed by Definition 5.1. Then, we can always add complete tuples to the relations in $d_1$ resulting, say in $d_2$, as long as $d_2 \models\!\approx \Sigma$ still holds. In this case, the closed world, in the form of $d_1$, can be viewed as the greatest lower bound of all open worlds emanating from $d_1$.

To conclude this brief survey we mention a recent approach taken by Keen and Rajasekar [KEEN93] who formalise the partitions arising from data dependencies such as FDs in the context of *rough sets* [PAWL82].

## 10 Concluding Remarks

We have investigated the additivity problem for NFDs and NINDs. The largest class of sets of NFDs for which additivity holds with respect to DB($\mathbf{R}$) is the class of monodependent sets of NFDs. In addition, checking whether a set of NFDs is monodependent can be done in time polynomial in the size of F. The largest class of sets of unary NINDs for which additivity holds with respect to DB($\mathbf{R}$) is the class of monodirectional sets of NINDs. Moreover, checking whether a set of unary NINDs is monodirectional can be done in time polynomial in the size of I. Finally, the largest class of sets of NFDs and unary NINDs for which additivity holds with respect to DB($\mathbf{R}$) is the class of monodependent sets of NFDs and monodirectional sets of unary NINDs which satisfy the cyclicity condition. Furthermore, checking whether a set of data dependencies, $\Sigma$, is in this last subclass can be done in time polynomial in the size of $\Sigma$.

In all of the situations when additivity is lacking this is due to some ambiguity in the specification of the set of data dependencies. In the case of NFDs either two sets of attributes determine the same attribute or there is some nontrivial cyclic behaviour between two NFDs. In the case of unary NINDs the left-hand side of a NIND is included in two distinct NINDs. Finally, in the case of NFDs and unary NINDs an attribute in the left-hand side of a NFD is also the left-hand side of a unary NIND. The subclasses of data dependencies which were shown to be additive resolve the ambiguity either by removing the offending data dependencies or by adding other data dependencies to the original set in order to neutralise the problem.

Suppose the user would like to maintain d $\models\!\approx \Sigma$ for all database states d. If $\Sigma$ is an additive set of data dependencies, then the database system need only check d $\models\!\approx \alpha$, $\forall \alpha \in \Sigma$. This can be done in time polynomial in the sizes of d, F and I. On the other hand, if $\Sigma$ is not an additive set of data dependencies the database system needs to check, in addition, that $\exists d' \in$ POSS(d) such that $\forall \alpha \in \Sigma, d' \models\!\approx \alpha$. It is not obvious that this can be done in polynomial time, since as was shown in the example following Proposition 4.2, SETOF(CHASE(d, F, I)) may contain an exponential number of databases and only one of these databases actually satisfies all the data dependencies in $\Sigma$.

We conjecture that the results obtained herein can be extended to the general class of sets of NFDs and NINDs with the aid of the observations made at the end of Sections 7 and 8. In any case, when the set of NINDs is not necessarily unary, it does not seem likely that there is a polynomial time algorithm for checking whether satisfaction is additive for a set of NINDs or a set of NFDs and NINDs due to the intractability of the respective implication problems for NINDs on their own and for NFDs and NINDs together. It would be interesting to investigate the additivity problem for null join dependencies, thus covering all the data dependencies that arise in practice. Finally, in [LEVE95b] we study the impact of assuming that sets of FDs are monodependent on normalisation theory in relational databases.

## References


[AHO79] A.V. Aho, C. Beeri and J.D. Ullman, The theory of joins in relational databases. *ACM Transactions on Database Systems* **4**, (1979), 297-314.

[ARMS74] W.W. Armstrong, Dependency structures of data base relationships. In: *Proceedings of the IFIP Congress*, Stockholm, pp. 580-583, 1974.

[ATZE84] P. Atzeni and N.M. Morfuni, Functional dependencies in relations with null values. *Information Processing Letters* **18**, (1984), 233-238.

[ATZE86] P. Atzeni and N.M. Morfuni, Functional dependencies and constraints on null values in database relations. *Information and Control* **70**, (1986), 1-31.

[ATZE93] P. Atzeni and V. De Antonellis, *Relational Database Theory.* Redwood City, Ca., Benjamin/Cummings, 1993.

[BEER79] C. Beeri and P.A. Bernstein, Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems* **4**, (1979), 30-59.

[BEER83] C. Beeri, R. Fagin, D. Maier and M. Yannakakis, On the desirability of acyclic database schemes. *Journal of the ACM* **30**, (1983), 479-513.

[BUNE91] P. Buneman, A. Jung and A. Ohori. Using powerdomains to generalize relational databases. *Theoretical Computer Science* **91**, (1991), 23-55.

[CASA84] M.A. Casanova, R. Fagin and C.H. Papadimitriou, Inclusion dependencies and their interaction with functional dependencies. *Journal of Computer and System Sciences* **28**, (1984), 29-59.

[CODD79] E.F. Codd, Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems* **4**, (1979), 379-434.

[COSM90] S.S. Cosmadakis, P.C. Kanellakis and M.Y. Vardi, Polynomial-time implication problems for unary inclusion dependencies. *Journal of the ACM* **37**, (1990), 15-46.

[DATE86] C.J. Date, Referential integrity. In: *Relational Database: Selected Writings.* Reading, Ma., Addison-Wesley, pp. 41-63, 1986.

[DATE93] C.J. Date and H. Darwen, *A Guide to the SQL Standard*, third edition. Reading, Ma., Addison-Wesley, 1993.

[DAVE90] B.A. Davey and H.A. Priestly, *Introduction to Lattices and Order.* Cambridge, U.K., Cambridge University Press, 1990.

[GARE79] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* New York, Freeman, 1979.

[GRAH91] G. Grahne, *The Problem of Incomplete Information in Relational Databases*, Lecture Notes in Computer Science, Volume 554. Berlin, Springer Verlag, 1991.

[GRAN80] J. Grant, Incomplete information in a relational database. *Fundamenta Informaticae* **3**, (1980), 363-378.

[HONE82] P. Honeyman, Testing satisfaction of functional dependencies. *Journal of the ACM* **29**, (1982), 668-677.

[IMIE83] T. Imielinski and W. Lipski Jr., Incomplete information and dependencies in relational databases. In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, San Jose, pp. 177-184, 1983.

[IMIE84] T. Imielinski and W. Lipski Jr., Incomplete information in relational databases. *Journal of the ACM* **31**, (1984), 761-791.

[JOHN84] D.S. Johnson and A. Klug, Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences* **28**, (1984), 167-189.

[KEEN93] D. Keen and A. Rajasekar, Rough sets and data dependencies. In: *Proceedings of the Workshop on Incompleteness and Uncertainty in Information Systems*, Montreal, pp. 87-101, 1993.

[LEVE92] M. Levene and G. Loizou, Inferring null join dependencies in relational databases. *BIT* **32**, (1992), 413-429.

[LEVE94a] M. Levene and G. Loizou , The additivity problem for functional dependencies in incomplete relations. Research Note RN/94/41, Department of Computer Science, University College London, 1994. To appear in *Acta Informatica.*

[LEVE94b] M. Levene and G. Loizou, Null inclusion dependencies in relational databases. Research Note RN/94/26, Department of Computer Science, University College London, 1994.

[LEVE95a] M. Levene and G. Loizou, A correspondence between variable relations and three-valued propositional logic. *International Journal of Computer Mathematics* **55**, (1995), 29-38.

[LEVE95b] M. Levene and G. Loizou, Database design for incomplete relations. Research Note RN/95/18, Department of Computer Science, University College London, 1995.

[LIBK91] L. Libkin, A relational algebra for complex objects based on partial information. In: *Proceedings of the Symposium on Mathematical Foundations of Database and Knowledge Base Systems*, Rostock, Germany, pp. 29-43, 1991.

[LIBK94] L. Libkin, A semantics-based approach to design of query languages for partial information. Research report MS-CIS-94-38. Computer and Information Science Department, School of Engineering and Applied Science, Univerity of Pennsylvania, 1994.

[LIEN79] Y.E. Lien, Multivalued dependencies with null values in relational databases. In: *Proceedings of the International Conference on Very Large Data Bases*, Rio de Janeiro, pp. 61-66, 1979.

[LIEN82] Y.E. Lien, On the equivalence of database models. *Journal of the ACM* **29**, (1982), 333-362.

[MAIE79] D. Maier, A.O. Mendelzon and Y. Sagiv, Testing implications of data dependencies. *ACM Transactions on Database Systems* **4**, (1979), 455-469.

[MAIE80] D. Maier, Minimum covers in the relational database model. *Journal of the ACM* **27**, (1980), 664-674.

[MANN92] H. Mannila and K.-J. Räihä, *The Design of Relational Databases.* Reading, Ma., Addison-Wesley, 1992.

[MITC83] J.C. Mitchell, The implication problem for functional and inclusion dependencies. *Information and Control* **56**, (1983), 154-173.

[PAWL82] Z. Pwalak, Rough sets. *International Journal of Computer and Information Sciences* **11**, (1982), 341-356.

[SCIO81] E. Sciore, Real world MVD's. In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, Ann Arbor, pp. 121-132, 1981.

[THAL89] B. Thalheim, On semantic issues connected with keys in relational databases permitting null values. *Journal of Information Processing Cybernetics* **25**, (1989), 11-20.

[ULLM88] J.D. Ullman, *Principles of Database and Knowledge-Base Systems*, Vol. I. Rockville, Md., Computer Science Press, 1988.

[VASS80] Y. Vassiliou, Functional dependencies and incomplete information. In: *Proceedings of the International Conference on Very Large Data Bases*, Montreal, pp. 260-269, 1980.