UNIVERSITÄT DORTMUND
Fachbereich Informatik

Lehrstuhl VIII
Künstliche Intelligenz

The Expanded Implication Problem of Data Dependencies[1]

LS–8 Report 16

Siegfried Bell

Informatik VIII
University Dortmund
44221 Dortmund Germany
bell@ls8.informatik.uni-dortmund.de

Dortmund, 12. June 1995

Universität Dortmund
Fachbereich Informatik

University of Dortmund
Computer Science Department

[1] Parts of this report had also been presented on the First International Conference on Knowledge Discovery in Databases (KDD 95), Montreal, Eds. U.M. Fayyad

Forschungsberichte des Lehrstuhls VIII (KI)
Fachbereich Informatik
der Universität Dortmund

ISSN 0943-4135

Anforderungen an:

Universität Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

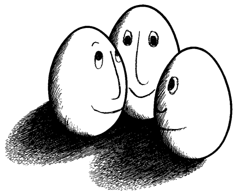
Research Reports of the unit no. VIII (AI)
Computer Science Department
of the University of Dortmund

ISSN 0943-4135

Requests to:

University of Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

e-mail: reports@ls8.informatik.uni-dortmund.de
ftp: ftp-ai.informatik.uni-dortmund.de:pub/Reports
www: http://www-ai.informatik.uni-dortmund.de/ls8-reports.html



The Expanded Implication Problem of Data Dependencies[2]

LS–8 Report 16

Siegfried Bell

Informatik VIII
University Dortmund
44221 Dortmund Germany
bell@ls8.informatik.uni-dortmund.de

Dortmund, 12. June 1995

Universität Dortmund
Fachbereich Informatik

[2]Parts of this report had also been presented on the First International Conference on Knowledge Discovery in Databases (KDD 95), Montreal, Eds. U.M. Fayyad

Abstract

The implication problem is the problem of deciding whether a given set of dependencies entails other dependencies. Up to now, the entailment of excluded dependencies or independencies is only regarded on a metalogical level, which is not suitable for an automatic inference process. But, the inference of independencies is of great importance for new topics in database research like knowledge discovery in databases.

In this paper, the expanded implication problem is discussed in order to decide entailment of dependencies and independencies. The main results are axiomatizations of functional, inclusion and multivalued independencies and the corresponding inference relations. Also, we discuss the use of independencies in knowledge discovery in databases and semantic query optimization.

1 Introduction

The implication problem in relational database theory can be defined as a test: Given a set of dependencies Σ and a dependency σ, find whether Σ entails σ, or $\sigma \in Cn(\Sigma)$. In order to avoid theorem proving in first order logic for testing entailment, there have been proposed more simple decision procedures, which are mainly based on axiomatizations of these dependencies.

But, new directions in database research require a more detailed point of view, because they deal for example with partial information. As a matter of fact knowledge discovery in databases is also concerned with the discovery of dependencies in databases. A typical situation can be described as follows: A set of dependencies Σ is known to be valid and a set of dependencies Σ' is known to be invalid in a database: What are the dependencies, which may also be valid or invalid in each of this relations, too ? Or, is a dependency σ already entailed by Σ and Σ' ? Since there are interactions between Σ and Σ', it is no longer sufficient to show $\sigma \in Cn(\Sigma)$ or $\sigma \notin Cn(\Sigma)$.

We would like to point out this view in respect with a description of the process of knowledge discovery in databases in general. Assume that S is the language of a particular class of constraints. $test(r, s)$ is a function which evaluates to $true$, if $s \in S$ and s holds in r, otherwise to *false*. The task of discovering a complete set of constraints which holds in r can be defined by: Find a set $V \subseteq S$ with $s \in V$ if and only if $test(r, s)$ evaluates to $true$. This definition can be transformed in a simple enumeration algorithm provided that S is finite for a given database:

$V := \{\}$
for each $s \in S$ **do**
  if $test(r, s)$ **then** $V := V \cup \{s\}$

Using this algorithm has the disadvantage that a lot of redundant constraints are tested. Therefore, we introduce a consequence relation $Cn(V)$ which computes all logical consequences of a set of constraints, i.e. $s \in Cn(V)$ if s is a consequence of V. The improved algorithm is:

$V := \{\}$
for each $s \in S$ **do**
  if $s \notin Cn(V)$
   then if $test(r, s)$ **then** $V := V \cup \{s\}$

But we still have unnecessary tests in our algorithm. For example, if we look at functional dependencies: Let be $A \to B \in V$ and $test(r, C \to B)$ evaluates to *false*. Thus, we do not have to test $C \to A$, because it cannot be valid in r. This shows that it is useful to have *negative* dependencies, because then we can avoid by the question $\neg s \notin Cn(V)$ this unnecessary test provided that S is still finite. The new algorithm is:

$V := \{\}$
for each $s \in S$ **do**
  if $s \notin Cn(V)$ and $\neg s \notin Cn(V)$
   then if $test(r, s)$ **then** $V := V \cup \{s\}$
         else $V := V \cup \{\neg s\}$

$\overset{n}{=}$	1	NULL	2
1	true	false	false
NULL	false	true	false
2	false	false	true

Figure 1: Equality operator $\overset{n}{=}$

In the following, we give solutions according functional independencies by an axiomatization and by an inference relation in the second and the third section. Then it is shown how functional dependencies can be maintained if new tuples are added or removed from the database. Unary inclusion independencies are investigated by an axiomatization and an inference relation in the next sections. In the seventh and eighth section the combination of both is discussed. A close relationship between multivalued dependencies and independencies is presented in the ninth section. Finally, we related our approach to others and discuss the work.

2 Functional Independencies

In this section we discuss functional independencies and their axiomatization. We assume familiarity with the definitions of relational database theory (for an overview see for example [Kanellakis, 1990]) and the basic properties of the classical consequence relation Cn. The capital letters $A, B, C, \ldots$ denote attributes, and X, Y, Z denote attribute sets. We do not distinguish between an attribute A and an attribute set $\{A\}$. Remember that every attribute is associated with a set of values, called its *domain*. Functional dependencies are defined as usual. Additionally, we consider `Null` values because the ISO standard permits `Null` values in any attribute of a *candidate key*. Therefore, we adopt a special equality operator for the definition of the FDs, $\overset{n}{=}$, which is illustrated in figure 1.

Now, we can define the consequences of a set of dependencies with: Let Σ be a set of functional dependencies, then $X \rightarrow Y$ is a consequence of Σ or $X \rightarrow Y \in CONS(\Sigma)$: whenever a relation satisfies Σ, then it satisfies $X \rightarrow Y$. This is the equivalent operator to the logical operator of classical logic. According to [Paredaens et al., 1989] a sound and complete axiomatization of FDs is given by one axiom schema and two inference rules:

Definition 1 *[Axiomatization of FDs] X, Y and Z are sets of attributes. An axiomatization of FDs is given by:*

$FD1 : (Reflexivity)$ *If* $X \subseteq Y$ *then* $Y \rightarrow X$

$FD2 : (Augmentation)$ *If* $W \subseteq V$ *then* $\frac{X \rightarrow Y}{XV \rightarrow YW}$

$FD3 : (Transitivity)$ $\frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z}$

Some well known rules likes union and complementation are logically implied by this system. The closure of attributes X regarding a set of FDs Σ is defined as: $closure(X, \Sigma) = \{Y | X \rightarrow Y \in CONS(\Sigma)\}$ and is denoted by $\overline{X}$.

Functional *independencies* have been introduced by Janas [Janas, 1988] to mirror functional dependencies. But they are meant for a totally different purpose: FIs are not semantical constraints on the data, but a support for the database designer in the task of identifying functional dependencies. In addition, they also help to improve the inference of functional dependencies.

In [Paredaens et al., 1989] *afunctional dependencies*[1] are introduced, but they are a sort of semantic constraints and much stronger than our functional independencies.

Definition 2 (Functional Independency (FI)) *$X \not\rightarrow Y$ denotes a functional independency. A relation r satisfies $X \not\rightarrow Y$ ($r \models X \not\rightarrow Y$), if there exist tuples t_1, t_2 of r with $t_1[X] \stackrel{n}{=} t_2[X]$ and $t_1[Y] \not\stackrel{n}{=} t_2[Y]$.*

The consequences of FDs and FIs are defined as follows: Let Σ be a set of FDs and Σ' a set of FIs. $Cn(\Sigma \cup \Sigma') := \{\sigma |$ for each relation r if $r \models \Sigma \cup \Sigma'$, then $r \models \sigma\}$, where σ is a FD or FI. $\Sigma \cup \Sigma'$ is called inconsistent, if there is no relation r with $r \models \Sigma \cup \Sigma'$.

An important property of the relationship between FDs and FIs is that the inference of FDs is not affected by the presence of FIs, stated by the first lemma.

Lemma 1 *Let Σ be a set of FDs, Σ' a set of FIs and $\Sigma \cup \Sigma'$ be consistent.*
$X \rightarrow Y \in Cn(\Sigma \cup \Sigma')$ if and only if $X \rightarrow Y \in Cn(\Sigma)$.

Proof: **(if)** is trivial by monotonicity of Cn.
(only-if) Assume that $X \rightarrow Y \notin Cn(\Sigma)$ and $X \rightarrow Y \in Cn(\Sigma \cup \Sigma')$. Then there must be a relation r with $r \models \Sigma$ and $r \not\models X \rightarrow Y$. This means, there are tuples t_1 and t_2 in r with $t_1[X] \stackrel{n}{=} t_2[X]$ and $t_1[Y] \not\stackrel{n}{=} t_2[Y]$. We can add for each element $V \not\rightarrow W \in \Sigma'$ two tuples which satisfy $V \not\rightarrow W$ by definition. Any values are assigned to the remaining attributes without affecting Σ, because $\Sigma \cup \Sigma'$ is consistent. Remember that our domains are countably infinite, which ensures that we can use new values if needed. The expanded relation satisfies Σ and $\Sigma \cup \Sigma'$ by construction, but not $X \rightarrow Y$, which is a contradiction.
$\square$

The next important observation is that FIs do not interact in the process of the inference. This can be illustrated by the fact that there exist at least two tuples for each FI but we cannot identify them and so we cannot reason about further consequences of FIs solely. For example, we cannot conclude from $X \not\rightarrow Y$ and $Y \not\rightarrow Z$ to the FI $X \not\rightarrow Z$. This can be clarified by the following observation:

Lemma 2 *Let Σ' be a set of FIs. If $X \not\rightarrow Y \in Cn(\Sigma')$, then there exists a FI $V \not\rightarrow W \in \Sigma'$ with $X \subseteq V$, and $W \cap Y \neq \{\}$.*

Proof: Assume that $X \not\rightarrow Y \in Cn(\Sigma')$, and $V \not\rightarrow W \notin \Sigma'$ with $X \subseteq V$, and $W \cap Y \neq \{\}$, and $\Sigma' = \{S_1 \not\rightarrow T_1, \ldots, S_n \not\rightarrow T_n\}$. We show by construction of two relations, that there exists a relation r so that if $r \models \Sigma'$, and $V \not\rightarrow W \notin \Sigma'$, then $r \not\models X \not\rightarrow Y$. Thus $X \not\rightarrow Y \notin Cn(\Sigma')$ which is a contradiction to the assumption.

[1] The definition of the AD $X \not\twoheadrightarrow Y$ requires that for each tuple t there exists a tuple t' so that $t[X] \stackrel{n}{=} t'[X]$ and $t[Y] \not\stackrel{n}{=} t'[Y]$.

The first relation r_1 with $2n$ tuples is constructed by assigning each attribute in each row pairwise different values, except for the attributes S_i with $S_i \not\rightarrow T_i \in \Sigma'$, the following condition holds $t_i[S_i] \stackrel{n}{=} t_{i+n}[S_i]$. Thus, $r_1 \models \Sigma'$.

The rows in the second relation r_2 with $2n$ tuples are assigned all the same value, except again for each $S_i \not\rightarrow T_i \in \Sigma'$, $t_i[S_i] \stackrel{n}{=} t_{i+n}[S_i]$ and $t_i[T_i] \not\equiv t_{i+n}[T_i]$. Thus, $r_2 \models \Sigma'$.

If $X \not\subseteq S_i$ for $i \in 1, \ldots, n$, then it is easy to see that $r_1 \models \Sigma'$, but $r_1 \not\models X \not\rightarrow Y$. This is a contradiction to the assumption, thus $X \subseteq S_i$.

Now, assume that $X \subseteq S_i$ for at least one i and $T_i \cap Y = \{\}$. It is easy to see that $r_2 \not\models X \not\rightarrow Y$, because if $t_i[X] \stackrel{n}{=} t_{i+n}[X]$, then also $t_i[Y] \stackrel{n}{=} t_{i+n}[Y]$. □

Corollary 1 *Let Σ be a set of FDs, Σ' a set of FIs.*

If $X \not\rightarrow Y \in Cn(\Sigma \cup \Sigma')$ then there exists a $R \not\rightarrow S \in \Sigma'$ such that $X \not\rightarrow Y \in Cn(\Sigma \cup \{R \not\rightarrow S\})$.

An axiomatization of FIs has already been given by Janas [Janas, 1988], which establishes an inference relation $\vdash_{\mathcal{J}anas}$.

Definition 3 (FIs) *The Axiomatization by Janas is given by:*

1. $\frac{X \not\rightarrow Y}{X \not\rightarrow YZ}$
2. $\frac{XZ \not\rightarrow YZ}{XZ \not\rightarrow Y}$
3. $\frac{X \rightarrow Y, X \not\rightarrow Z}{Y \not\rightarrow Z}$

We show with a counterexample that this inference relation is not complete, i.e., there exists some $X \not\rightarrow Y$ with $X \not\rightarrow Y \in Cn(\Sigma \cup \Sigma')$ and $\Sigma \cup \Sigma' \not\vdash_{\mathcal{J}anas} X \not\rightarrow Y$.

Lemma 3 *The following inference rule is correct:*

$$\frac{X \rightarrow Y, Z \not\rightarrow Y}{Z \not\rightarrow X}$$

Proof: trivial by assuming that the conclusion is not satisfied and Armstrong's Axioms. □

Lemma 4 $\{X \rightarrow Y, Z \not\rightarrow Y\} \not\vdash_{\mathcal{J}anas} Z \not\rightarrow X$.

Proof: We assume that X, Y and Z are disjoint. Then it is obvious that the first and the second rule cannot have be applied to infer $Z \not\rightarrow X$. Thus, the third rule can be applied only. But Z is not in the closure of Y and Σ, i.e. $\{X \rightarrow Y\} \not\vdash_A Y \rightarrow Z$. Thus, $Z \not\rightarrow X$ cannot be inferred. □

Corollary 2 *The axiomatization by Janas is not complete.*

Instead, we propose the following axiomatization:

Definition 4 (Inference of FIs) *An inference relation $\vdash_{fi}$ is given by an axiomatization of the FDs and the following inference rules:*

$$FI1: \quad \frac{V \not\to YU, U \subseteq V}{V \not\to Y}$$

$$FI2: \quad \frac{X \to Y, X \not\to Z}{Y \not\to Z}$$

$$FI3: \quad \frac{Y \to Z, X \not\to Z}{X \not\to Y}$$

For example, the functional independency $X \not\to YZ$, which is a consequence of Janas's first inference rule, can be inferred by $\vdash_{fi}$ as follows: we infer $YZ \to Y$ by Armstrong's Axiom and use FI3 to infer the FI $X \not\to YZ$ from $X \not\to Y$ and $YZ \to Y$. FI1 reflects lemma 2, because FIs can be inferred from a set of FIs only by this rule.

Theorem 1 (Soundness) *The inference rules of definition 4 are correct.*

Proof: (Soundness) By Lemma 1 it is sufficient to show the soundness of FI1, FI2 and FI3 w.r.t. functional independencies:

- (FI1) We have to show that $V \not\to Y \in Cn(\{V \not\to YU\})$. This means that each relation that satisfies $V \not\to YU$ must satisfy $V \not\to Y$. By definition there are tuples t_1, t_2 with $t_1[V] \stackrel{n}{=} t_2[V]$ and $t_1[YU] \not\stackrel{n}{=} t_2[YU]$. Since $U \subseteq V$, it follows that $t_1[U] \stackrel{n}{=} t_2[U]$, thus $t_1[Y] \not\stackrel{n}{=} t_2[Y]$ and $t_1[V] \stackrel{n}{=} t_2[V]$ and therefore $V \not\to Y$.
- (FI2) $\{X \to Y, X \not\to Z\}$ means there are tuples t_1, t_2 with $t_1[X] \stackrel{n}{=} t_2[X]$, $t_1[Z] \not\stackrel{n}{=} t_2[Z]$ and for all tuples, particularly for t_1, t_2 $t_1[X] \stackrel{n}{=} t_2[X]$ and $t_1[Y] \stackrel{n}{=} t_2[Y]$. Then $t_1[Y] \stackrel{n}{=} t_2[Y]$ and $t_1[Z] \not\stackrel{n}{=} t_2[Z]$ and therefore $Y \not\to Z$.
- (FI3) see Lemma 3 □

Theorem 2 (Completeness) *The inference rules of definition 4 are complete.*

Before, we state the theorem we need a further definition about FDs.

Definition 5 (Base of FDs) *Let Σ a set of FDs and X a set of attributes. Then the base of X is defined by:*

$$\underline{X} = \{Y \mid Y \to X \in Cn(\Sigma)\}$$

Note, that the base of a set of attributes is a set of sets. To prevent too many sets, we restrict the set to minimal sets. Thus, $\underline{X}$ consists of minimal sets S_i, or $\underline{X} = \{S_i \mid 1 \leq i \leq n\}$ and if $S_j \supseteq S_i$ then $S_j = S_i$ for $i, j \in 1, \ldots, n$. For example, let $\Sigma = \{AB \to C, CD \to EF\}$ be a set of FDs, then $\underline{EF} = \{ABD, CD, EF\}$. Now, we can prove the completeness:

The idea of the proof is, that it is possible to partition all subsets of U into the set of all S_i (and their subsets), and in $V = \{V_j |$ for all S_i it holds, that $V_j \in P^U \backslash S_i$ and $V_j \not\subseteq S_i\}$. There are only two possibilities for the set Y, which will be used in the following proof:

1. $Y \subseteq S_i$
2. $Y \supseteq V_j$

Proof: It is sufficient to show the completeness w.r.t. functional independencies by lemma 1. Let Σ be a set of FDs, Σ' a set of FIs and $\Sigma \cup \Sigma'$ consistent. We show that if $X \not\rightharpoonup Y \in Cn(\Sigma \cup \Sigma')$ then $\Sigma \cup \Sigma' \vdash_{fi} X \not\rightharpoonup Y$. By corollary 1 we know that $X \not\rightharpoonup Y \in Cn(\Sigma \cup \{R \not\rightharpoonup S\})$ for $R \not\rightharpoonup S \in \Sigma'$. Remember that $\overline{R}$ is the closure of R w.r.t. Σ. Then we can partition the attributes of each relation which satisfy $\Sigma \cup \{R \not\rightharpoonup S\}$ into two disjunctive sets: $\overline{R}$, and the set U of the remaining attributes. The existence of $\overline{R}$ is guaranteed by definition.

Now we take a look at the following cases and show how we can use our inference rules:

- Assume that $X \not\subseteq \overline{R}$: Then we can easily construct a relation r with two rows which satisfies $\Sigma \cup \{R \not\rightharpoonup S\}$ but not $X \not\rightharpoonup Y$ by assigning each attribute of U different values. If the left hand sides of the FDs in Σ are in $\overline{R}$, then also the right hand sides. We assume that 0 and 1 are in the domain of the attributes. Then there are no tuples t_1, t_2 with $t_1[X] \overset{n}{=} t_2[X]$. Thus, $r \not\models \Sigma \cup \{R \not\rightharpoonup S\}$, but $r \models X \not\rightharpoonup Y$. Therefore $X \subseteq \overline{R}$.

U	$\overline{R}$
$0\dots$	$0\dots$
$1\dots$	$0\dots$

- Let $X \subseteq \overline{R}$, $\underline{S} = S_i$ for $1 \leq i \leq n$, and $Y \cap S_i = Z$ for an i:

  1. $Z = Y$: This means there is a set of attributes W, which is disjunct to Y and $YW \rightharpoonup S$. By repeated application of $FI2$ we know that $\overline{R} \not\rightharpoonup S$, and by $FI3$ that $\overline{R} \not\rightharpoonup YW$. There are two case:
     (a) $W \subseteq \overline{R}$: Thus, by $FI1$ $\overline{R} \not\rightharpoonup Y$ is inferred and by $FI2$ $X \not\rightharpoonup Y$.
     (b) $W \not\subseteq \overline{R}$: Thus, we infer by $FI1$ the FI $YW' \rightharpoonup S$ with the attributes $W' = W \backslash \overline{R}$ and construct a counterrelation r by

$\overline{R}$	W'	S	$U \backslash S \cup W$
$0\dots$	$0\dots$	$0\dots$	$0\dots$
$0\dots$	$1\dots$	$1\dots$	$1\dots$

     Then it is easy to see that $r \models \Sigma \cup \{R \not\rightharpoonup S\}$, but $r \not\models X \not\rightharpoonup Y$, which is a contradiction to the assumptions.
  2. $Z = \{\}$: This means, that a relation r can be constructed with a partition of all attributes into $\overline{R}$, all attribute $\underline{S}$ which occur in a S_i and the remaining attributes U:

$\overline{R}$	$\underline{S} \backslash \overline{R}$	$U \backslash \underline{S}$
$0\dots$	$0\dots$	$0\dots$
$0\dots$	$1\dots$	$0\dots$

     Then it is easy to see that $r \models \Sigma \cup \{R \not\rightharpoonup S\}$ but $r \not\models X \not\rightharpoonup Y$, which is a contradiction. Thus, $Z \neq \{\}$.
  3. $Z = S_i$: By assumption, we know that $Y \rightharpoonup S$ and $R \not\rightharpoonup S$. By rule $FI3$ we conclude that $R \not\rightharpoonup Y$. We know also, that $X \subseteq \overline{R}$, respectively, that $R \rightharpoonup X$. Thus, by $FI2$ we conclude that $X \not\rightharpoonup Y$.

4. $Z \subseteq \overline{R}$: Here the same counterrelation can be constructed as in case 2.
5. $Z \subseteq U$: The same as in case 2. □

Corollary 3 (Completeness and Soundness) *Let Σ and Σ' be a set of FDs and FIs respectively, and $\Sigma \cup \Sigma'$ be consistent. $X \not\rightarrow Y \in Cn(\Sigma \cup \Sigma')$ if and only if $\Sigma \cup \Sigma' \vdash_{fi} X \not\rightarrow Y$.*

3 Inference of Functional Independencies

Our system consists of three elements: initialization, entailment, and verification. It is roughly sketched in table 1. First, we initialize our data structure `List` for the FDs and FIs. Then, we generate hypothetical dependencies, check if these are already entailed by the known dependencies or independencies, and verify the remaining ones by querying the database. We use a kind of breath first search because we generate only hypotheses which are not related by each other. Terminating is ensured, because if no already entailed hypotheses can be generated, then the algorithm stops.

3.1 Verification

Functional dependencies can be verified by sorting the tuples of the relation which takes $O(n\ log\ n)$ time w.r.t. the number of tuples, cf. [Mannila and Räihä, 1991]. In our implementation we use the `nvl` statement in SQL to handle the NULL values. We refer the reader to our previous work in [Bell and Brockhausen, 1995] for details on the verification of dependencies.

3.2 Entailment

Entailment of FDs is often discussed by studying if $\Sigma \vdash_A X \rightarrow Y$ holds where Σ is a set of FDs and $n = |\Sigma|$. This can be decided in linear time with appropriate data structures [Kanellakis, 1990]. By Lemma 2, we can easily construct an algorithm for testing FIs:

function $\Sigma \cup \Sigma' \vdash_{fi} X \not\rightarrow Y$;
begin
  for each $V \not\rightarrow W \in \Sigma'$ **do**
    for each $Z \subseteq W \cap V$ **do**
      if $\Sigma \vdash_A V \rightarrow X$ **and** $\Sigma \vdash_A Y \rightarrow W \backslash Z$
       then return Yes;
return No;
end;

It is obvious that testing FIs takes $O(m \cdot n^2)$ time where $n = max(\Sigma, \Sigma')$ and m is the number of attributes. If we demand that for each $S_i \not\rightarrow T_i \in \Sigma'$ and $S_i \cap T_i = \{\}$, then testing takes $O(n^2)$ time. Correctness and completeness follow immediately from the previous section.

The sets of FDs and FIs are usually very large. We can reduce these sets taking into account the following observation: The set of functional dependencies is partitioned into equivalence classes by the satisfiability definition. Each class of functional dependencies

1. Initialize `List`.
2. Repeat
   (a) Take an element t from `List` and generate all tuples T with a fixed length that are not entailed by `List`.
   (b) Query DB server for T.
   (c) Add T to *List* and find a minimal cover for it.
3. until no already entailed hypothesis can be generated

Table 1: Description of our System

specifies the same set of admissible relations. As these equivalence classes will typically contain a large number of elements, it is reasonable to define a suitable representation with a minimal number of elements. This representation is usually called a minimal cover, see [Maier, 1980]. We can simply extend the definitions in [Maier, 1980] by using our inference relation $\vdash_{fi}$:

Definition 6 (Minimal Cover) *Let* Σ *be a minimal set of FDs.*
Σ' *is a set of FIs and is called minimal if for all* $V \not\rightarrow W \in Cn(\Sigma \cup \Sigma')$ *there exists no* $X \not\rightarrow Y \in \Sigma'$ *with* $\Sigma \cup \Sigma' \backslash \{X \not\rightarrow Y\} \vdash_{fi} V \not\rightarrow W$.

Therefore, minimizing can be done by repeated application of $\vdash_{fi}$ and takes $O(n^3)$ time for some set of FDs and FIs.

3.3 Initialization

We initialize our data structure with information about primary keys and sufficient conditions for FIs as given by the cardinality of attributes. Cardinality Dependencies (CDs) was introduced by Kanellakis et al. [Kanellakis et al., 1983] for an axiomatization of unary inclusion dependencies and FDs in the finite case of databases. CDs simply compare the numbers of different values of an attribute in a certain relation. They propose the following rules for the interaction of CDs and FDs which we extend to sets of attributes. The fact that the cardinality of the values in X is greater or equal than Y is expressed by the CD $|X| \geq |Y|$. We omit the inference relation of the operator " $\geq$ " which is given by the corresponding relation $\geq$ and characterized by $\vdash_c$.

Definition 7 (Interaction of CDs and FDs) *Let* X *and* Y *be sets of attributes. The interaction of CDs and FDs is given by two inference rules :*

$$\frac{X \rightarrow Y}{|X| \geq |Y|} \qquad\qquad \frac{X \rightarrow Y, |Y| \geq |X|}{Y \rightarrow X}$$

Theorem 3 (Soundness of $\vdash_{fc}$) *The inference rules of definition 7 are sound.*

Proof: (soundness) trivial $\square$

For completeness we need a definition of CDs of a set of FDs and the following lemmas. f_{CD} is a function that transforms a set of FDs in a set of CDs so that simply $\rightarrow$ is replaced by $\geq$, for example $f_{CD}(\{X \rightarrow Y\}) = \{|X| \geq |Y|\}$. The first lemma states that if a CD is inferred from a set of CDs and FDs, then it can be inferred from a set of CDs.

Lemma 5 (Interaction 1) *Let Σ and Γ be a set of FDs and Γ CDs respectively. $|X| \geq |Y| \in Cn(\Sigma \cup \Gamma)$ if and only if $|X| \geq |Y| \in Cn(\Gamma \cup f_{CD}(Cn(\Sigma)))$.*

Proof: (if) is trivial by the monotonicity of Cn and the definition of f_{CD} which reflects the first rule
(only-if:) We assume that $|X| \geq |Y| \notin Cn(\Gamma \cup f_{CD}(Cn(\Sigma)))$ and $|X| \geq |Y| \in Cn(\Sigma \cup \Gamma)$. But this cannot be the case because f_{CD} is bijective and Cn is idempotent. $\square$

The second lemma states that if a FD is inferred from a set of CDs and FDs, then it can be inferred from the set of FDs solely, or a kind of inverse of it can be inferred from the set of FDs.

Lemma 6 (Interaction 2) *Let Σ be a set of FDs and Γ a set of CDs. If $X \rightarrow Y \in Cn(\Sigma \cup \Gamma)$ then $X \rightarrow Y \in Cn(\Sigma)$ or $Y \rightarrow X \in Cn(\Sigma)$.*

Proof: (Case 1) if $X \rightarrow Y \in Cn(\Sigma \cup \Gamma)$ then $X \rightarrow Y \in Cn(\Sigma)$ is trivial.
(Case 2) Thus, $X \rightarrow Y \notin Cn(\Sigma)$ and $X \rightarrow Y \in Cn(\Sigma \cup \Gamma)$. We assume that $Y \rightarrow X \notin Cn(\Sigma)$. Hence, there is at least a relation r so that $r \models \Sigma$, but $r \not\models Y \rightarrow X$. The following relation r satisfies each element of Σ, but does not satisfy neither $X \rightarrow Y$ nor $Y \rightarrow X$. Again $\overline{X}$ is the closure of X, respectively $\overline{Y}$ of Y, and U is the set of the remaining attributes. We can see this by filling the second row of U. If $W \subseteq U$ and $W \rightarrow X$ and $W \rightarrow Y$, then assign 2, else if $W \rightarrow X$, then assign 0 else 1 to W. The remaining attributes can be assigned the value 1, since all CDs are satisfied w.r.t. $Cn(\Gamma)$. Other cases cannot arise because $\Sigma \cup \Sigma'$ is consistent. For example, if $W \rightarrow X$ and $W \rightarrow Y$ and $|X| \geq |W|$, then we can infer by the second inference rule $X \rightarrow W$, but this is a contradiction because we can also infer $X \not\rightarrow W$ by $\vdash_{fi}$. Therefore, we have a relation r with $r \models \Sigma \cup \Gamma$ and $r \not\models X \rightarrow Y$ which is a contradiction to the assumption.

U	$\overline{X}$	$\overline{Y}$
$0\ldots$	$0\ldots$	$0\ldots$
$\ldots$	$0\ldots$	$1\ldots$
$1\ldots$	$1\ldots$	$1\ldots$

Now we are able to prove the completeness of $\vdash_{fc}$:

Theorem 4 (Completeness of $\vdash_{fc}$) *Let Σ be a set of FDs and Γ a set of CDs. An inference relation $\vdash_{fc}$ is given by Armstrong's Axioms, an axiomatization of $\geq$ and the inference rules of definition 7.*

Proof: First, we show that if $|X| \geq |Y| \in Cn(\Sigma \cup \Gamma)$, then $\Sigma \cup \Gamma \vdash_{fc} |X| \geq |Y|$. By Lemma 5 it is sufficient to show that if $|X| \geq |Y| \in Cn(\Sigma \cup \Gamma)$ then $\Gamma \cup f_{CD}(Cn(\Sigma)) \vdash_{fc} |X| \geq |Y|$. This is trivial by the axiomatization of $\geq$.

In the second part of the proof we have to show that if $Y \rightarrow X \in Cn(\Sigma \cup \Gamma)$, then $\Sigma \cup \Gamma \vdash_{fc} Y \rightarrow X$. Without loss of generality we assume that $Y \rightarrow X \notin Cn(\Sigma)$. Otherwise we can use the completeness of Armstrong's Axioms. First, we show that $|Y| \geq |X| \in Cn(\Sigma \cup \Gamma)$. If $|Y| \geq |X| \notin Cn(\Sigma \cup \Gamma)$, then there must be a r with $r \models \Sigma \cup \Gamma$, $r \not\models |Y| \geq |X|$ and $r \models |Y| \geq |X|$ due to the soundness of the first rule. Therefore, $|Y| \geq |X| \in Cn(\Sigma \cup \Gamma)$. Since the second interaction Lemma we know that $X \rightarrow Y \in Cn(\Sigma)$. Therefore, we can apply the second inference rule and conclude that $\Sigma \cup \Gamma \vdash_{fc} Y \rightarrow X$. □

We also use CDs to initialize the data structures of the FIs by the following inference rules. We do this by the introduction of a stronger form of CDs, $|X| > |Y|$, where $>$ has the obvious meaning and we expand $\vdash_c$ adequately.

Definition 8 (Interaction of CDs and FIs) *Let X and Y be sets of attributes. An axiomatization of FDs, FIs and CDs is given by the definition 7, and an axiomatization of FDs and FIs, and the following two inference rules.*

$$\text{CD-FI1: } \frac{|X| \geq |Y|,\ X \not\rightarrow Y}{Y \not\rightarrow X} \qquad\qquad \text{CD-FI2: } \frac{|X| > |Y|}{Y \not\rightarrow X}$$

The following lemma shows that we have to regard FIs during the inference procedure only:

Lemma 7 (Interaction of FDs, FIs and CDs) *Let Σ, Σ' and Γ be sets of FDs, FIs and CDs and $\Sigma \cup \Sigma' \cup \Gamma$ be consistent. $X \rightarrow Y \in Cn(\Sigma \cup \Sigma' \cup \Gamma)$ if and only if $X \rightarrow Y \in Cn(\Sigma \cup \Gamma)$.*

Proof: (if) is trivial.
(only–if) Assume $X \rightarrow Y \in Cn(\Sigma \cup \Sigma' \cup \Gamma)$ and $X \rightarrow Y \notin Cn(\Sigma \cup \Gamma)$, then there must be at least one relation r with $r \models \Sigma \cup \Gamma$ and $r \not\models X \rightarrow Y$. Therefore, there are tuples t_1, t_2 with $t_1[X] = t_2[X]$ and $t_1[Y] \neq t_2[Y]$. By consistency we can expand this relation r by adding new tuples, two for each element of Σ' so that $r \models \Sigma \cup \Sigma' \cup \Gamma$ and $r \models \Sigma \cup \Gamma$ which is a contradiction. (The construction is done in a similar way as above.) □

Now an inference relation can be defined based on FDs, FIs and CDs which clarifies the usefulness of FIs.

Theorem 5 *Definition 8 establishes an inference relation $\vdash_i$ which is sound and complete.*

Proof: (Soundness) is trivial.

Let Σ, Σ' and Γ be sets of of FDs, FIs and CDs and $\Sigma \cup \Sigma' \cup \Gamma$ is consistent. We have to show that $X \not\rightarrow Y \in Cn(\Sigma \cup \Sigma' \cup \Gamma)$ implies $\Sigma \cup \Sigma' \cup \Gamma \vdash_i X \not\rightarrow Y$. By definition we know that each relation r which satisfies $\Sigma \cup \Sigma' \cup \Gamma$ has to satisfy $X \not\rightarrow Y$. Thus, we have to regard only three cases for the cardinalities of X and Y in r by Lemma 7: first, if $|Y| > |X|$, then by the first inference rule we infer $X \not\rightarrow Y$; second, if $|X| = |Y|$, then we know that two tuples t_1, t_2 must be in r with $t_1[X] = t_2[X]$ and $t_1[Y] \neq t_2[Y]$, because $X \not\rightarrow Y$ holds, and the FDs and FIs can be completely inferred. Hence, to adjust the

A_1	A_2	A_3	A_4	...	A_{2m-1}	A_{2m}	A_{2m+1}
0	1	0	$2\ 2^m+1$	...	0	$2(m-1)2^m$	0
0	2	0	$2\ 2^m+2$	...	0	$2(m-1)2^m+1$	1
3	0	0	$2\ 2^m+3$	...	0	$2(m-1)2^m+2$	0
4	0	0	$2\ 2^m+4$	...	0	$2(m-1)2^m+3$	1
...							...
...							...
0	$2\ 2^m-3$	$4\ 2^m-3$	0	...	$2m2^m-3$	0	0
0	$2\ 2^m-2$	$4\ 2^m-2$	0	...	$2m2^m-2$	0	1
$2\ 2^m-1$	0	$4\ 2^m-1$	0	...	$2m2^m-1$	0	0
$2\ 2^m$	0	$4\ 2^m$	0	...	$2m2^m$	0	1

Table 2: Example Relation for FIs

cardinalities there must be also two tuples t_3, t_4 with $t_3[Y] = t_4[Y]$ and $t_3[X] \neq t_4[X]$. This implies that $Y \not\rightarrow X \in Cn(\Sigma \cup \Sigma' \cup \Gamma)$. Then we can infer with the first inference rule that $\Sigma \cup \Sigma' \cup \Gamma \vdash_i X \not\rightarrow Y$. Third, if $|X| > |Y|$, then we can infer $Y \not\rightarrow X$ and also with the second inference rule $X \not\rightarrow Y$. □

Unfortunately, it turned out that testing CDs by our SQL-interface is as much expensive as testing FDs[2]. Thus, we check the cardinality of unary CDs in one pass only and approximate CD from them by the following Lemma:

Lemma 8 *Let A be the attribute with the maximal cardinality in X and $Y = B_1, \ldots, B_n$. If $|A| \geq (|B_1| \ldots |B_n|)$, then $|X| \geq |Y|$.*

Proof: trivial □

So, we initialize our data structures of FDs and FIs with the in this manner approximated CDs and the inference rule CD-FI2 only. But it is easy to see, that the number of CDs grows exponentially w.r.t. to the number of attributes. Therefore, this algorithm is in EXPTIME.

3.4 Complexity of the System

Our system is in EXPTIME because there exist relations with the number of FDs in a minimal cover growing exponentially w.r.t. the number of attributes. This has been shown by Beeri et al. [Beeri et al., 1984] and also by Mannila and Räihä [Mannila and Räihä, 1991]. As there are relations with the number of FIs growing exponentially the performance cannot be improved by using FIs instead of FDs.

Theorem 6 (Cardinality of the set of FIs) *For each n there exists a relation r which satisfies a minimum cover of FIs with the cardinality $\Omega(2^{n/2})$.*

Proof: We construct a relation r and show that the cardinality of the minimum cover of FIs Σ' grows exponentially w.r.t. the number of the attributes of r, see table 2. Assume without loss of generality that $n = R = 2m+1$. Then we add for each $1 \leq k \leq 2^m$ two

[2] We can only count single attributes by the count – statement in SQL.

tuples t and t' where X contains exactly one of the attributes A_{2i-1} and A_{2i} for $1 \leq i \leq 2m$ with $t[X] = 0$, $t'[X] = 0$, $t[A_{2m+1}] = 0$ and $t'[A_{2m+1}] = 1$. Each attribute not mentioned in each t, t' gets a new number as value.

Now we define $\Sigma' = X \not\rightarrow A_{2m+1}$ such that for each i, where $1 \leq i \leq 2m$, X contains exactly one of the attributes A_{2i-1} and A_{2i}. First, we show that r satisfies Σ'. We can find for each $X \not\rightarrow A_{2m+1}$ two tuples t_1 and t_2 with $t_1[X] = t_2[X]$ but $t_1[A_{2m+1}] \neq t_2[A_{2m+1}]$ by construction of r. Second, we show that there exists no X' with $X \subset X'$ and $X' \not\rightarrow A_{2m+1}$. This can be seen by the fact that each $A_j \notin X$ has a unique value. Hence, $X' \rightarrow A_{2m+1}$ and Σ' is a minimum cover. $\square$

If the relation of Mannila and Räihä is added to ours, then it is easy to see that relations exist where the sets of FDs and FIs grow exponentially w.r.t. the number of attributes. Again, we argue that our goal, to minimize database access, can be achieved with this system.

4 Maintenance of FDs

Obviously, the discovered FDs can become invalid, because they only describe the current state of the database. Therefore, the discovered FDs have to be maintained if new tuples are added, old tuples are deleted, or existing tuples are updated.

If maintenance of FDs is seen as revision, it is more suitable to do theory revision than base revision as introduced by Gärdenfors [Gärdenfors, 1988]. In contrast to theory revision, base revision works on the whole consequence set So it is easy to see, for example, that by adding a new tuple the second FD of the set $\{AB \rightarrow CD, CD \rightarrow EF\}$ may become invalid, but the $AB \rightarrow EF$ remains valid. Therefore, in a first step the minimal set of FDs Σ is transformed into a set Σ_m of FDs. This new set is called *most general* and can be computed with the following algorithm:

for each $X \rightarrow Y \in \Sigma$ **do**
  if $Y = closure(X, \Sigma) \backslash X$ **then**
    $\Sigma_m := \Sigma_m \cup \{X \rightarrow Y\}$

Obviously, the complexity depends on the cardinality of Σ and the closure operation which is mentioned above.

4.1 Inserting Tuples

If new tuples are added, FDs may become invalid. Thus, each FD is checked if it is still valid. If not then the FD has to be replaced by a set of FDs which are valid. The algorithm is listed in table 3 and is applied before the tuple is inserted. Let $(d_1, \ldots, d_{n+m+l})$ be the new tuple, r the corresponding relation with the relation scheme $R = (A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_l)$, $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ the selected FD and $C_1, \ldots, C_l$ the remaining attributes. The c_i can consist of values and that we expand the left hand side of each invalid FD by attributes which values are different from the selected ones.

The advantage of this algorithm is that it only one simple query for each new tuple and for each FD is needed. It is easy to see that the removed FDs are no longer valid. For the correctness, we first give the following lemma:

$\Sigma_n := \{\}$
for each $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \in \Sigma$ **do**
begin
  $b_1, \ldots, b_m, c_1, \ldots, c_l :=$
    select $B_1, \ldots, B_m, C_1, \ldots, C_l$
    from r
    where $A_{i_1} = a_{i_1}, \ldots, A_{i_n} = a_{i_n}$
  if $\bigwedge_{i=0,\ldots,m} b_i = d_{n+i}$ then
   $\Sigma_n := \Sigma_n \cup A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$
  else begin
   if there is a minimal ν with $C_{l_1}, \ldots, C_{l_\nu}$ and $\bigwedge_{i=1,\ldots,\nu} d_{l_\nu} \neq c_{l_\nu}$ **then**
   $\Sigma_n := \Sigma_n \cup A_1, \ldots, A_n, C_{l_1}, \ldots, C_{l_\nu} \rightarrow B_1, \ldots, B_m$
  end
end

Table 3: Algorithm for Inserting New Tuples

Lemma 9 *Each generated FD is valid.*

Proof: We call the new tuple t_1. We know that $A_1, \ldots, A_n, B_1, \ldots, B_m$ is invalid. Then there must be at least one tuple t_2 so that $t_1[A_1, \ldots, A_n] \stackrel{n}{=} t_2[A_1, \ldots, A_n]$ and $t_1[B_1, \ldots, B_m] \not\equiv t_2[B_1, \ldots, B_m]$. We know by construction that $t_1[C_{l_1}, \ldots, C_{l_v}] \not\equiv t_2[C_{l_1}, \ldots, C_{l_v}]$. Hence, $t_1[X, C_{l_1}, \ldots, C_{l_v}] \not\equiv t_2[X, C_{l_1}, \ldots, C_{l_v}]$ and $A_1, \ldots, A_n, C_{l_1}, \ldots, C_{l_v} \rightarrow B_1, \ldots, B_m$ is valid. □

Completeness can now be seen by the following lemma:

Lemma 10 *Let Σ be the former set of FDs and Σ_n be the revised set of FDs. r_n is obtained by expanding r by one tuple. Assume that $\Sigma \models X \rightarrow Y$ and $r \models X \rightarrow Y$. If $\Sigma_n \not\models X \rightarrow Y$, then $r_n \not\models X \rightarrow Y$.*

Proof: By correctness we only add valid FDs. By minimality of the added attributes to the LHS of the FD, it is guaranteed that an FD with less attributes at the LHS is not valid. □

We conclude that if our algorithm is applied on Σ and the result is Σ', then $r' \models \Sigma'$ by the lemmas.

4.2 Deleting Tuples

Deleting tuples does not affect the old set of FDs, but some new FDs may become valid. Therefore, we have to revise the FDs and add new FDs if necessary. Unfortunately it turned out, that deleting tuples is the same as discovering FDs with a given starting set of FDs. Therefore computation in this case can be as expensive as the discovery process.

4.3 Updating Tuples

Normally, updating tuples can be seen as a combination of deleting and inserting tuples. But sometimes we can simplify this process by comparing the old values with the new ones.

Assume that $d = (a_1, \ldots, a_n, b_1, \ldots, b_m, c_1, \ldots, c_l)$ is the tuple which will be updated by the values $(a'_1, \ldots, a'_n, b'_1, \ldots, b'_m, c'_1, \ldots, c'_l)$, and the selected FD is $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$.

- If the values of the attributes of the left and the right hand side do not change, then we have nothing to do.
- If $\bigwedge_{i=1,\ldots,m} b_i = b'_i$ and at least one value of A_i, $1 \leq i \leq n$, does not occur in r, then we have only to apply the algorithm for deleting tuples.

5 Unary Inclusion Independencies

Another important class of dependencies are the so called unary inclusion dependencies. We regard only unary inclusion dependencies, because there exists an axiomatization regarding functional dependencies. Originally, the concept of inclusion dependencies in relational database theory has been a generalization of Codd's notion of a *foreign key*. Unary inclusion dependencies are restricted to only one attribute. Again, we denote a set of formulas consistent if there exists a database which satisfies the set. Satisfiability is defined as in the second section.

Definition 9 (Unary Inclusion Dependencies (UIND)) *Let A be an attribute of the relation scheme R, and B of S, and a an element of the domain of B. $R[A] \supseteq S[B]$ is an UIND: if $a \in S[B]$, then $a \in R[A]$.*

The interaction of UINDs can be formally described by an axiomatization. According to [Kanellakis, 1990] a sound and complete axiomatization for unary inclusion dependencies (UIND) is given by the following definition:

Definition 10 (Inference of Unary Inclusion Dependencies) *Let A be an attribute of the relation scheme R, and B of S, and C of T. Inference rules of unary inclusion dependencies are given by:*

$$U1 : (Reflexivity) \quad R[A] \supseteq R[A]$$

$$U2 : (Transitivity) \quad \frac{R[A] \supseteq S[B], S[B] \supseteq T[C]}{R[A] \supseteq T[C]}$$

The concept of unary inclusion independencies (UINIs) simplifies the discovery of UINDs. Therefore, UINDs are introduced and the interaction is described.

Definition 11 (Unary Inclusion Independencies (UINI)) *Let A be an attribute of the relation scheme R, and B of S, and a an element of the domain of B. $R[A] \not\supseteq S[B]$ is an UINI: if there exists an a with $a \in S[B]$ and $a \notin R[A]$.*

Now, it will be shown that there is only an interaction from UINDs to UINIs and not vice versa. Therefore, we denote by I a set of UINDs and by I' a set of UINIs.

Lemma 11 *If $I \cup I'$ is consistent then $R[A] \supseteq S[B] \in Cn(I \cup I')$ iff $R[A] \supseteq S[B] \in Cn(I)$*

Proof: **(if)** is trivial by the monotonicity of $\models$.
(only-if) We know that $R[A] \supseteq S[B] \in Cn(I \cup I)$ and $I \cup I'$ is consistent. If $R[A] \not\supseteq S[B] \in I'$ then $I \cup I'$ is inconsistent. But this is not the case, therefore it exists at least one database which satisfies $I \cup I'$ and for each database db it is the case that $db \models R[A] \supseteq S[B]$.

Assume that $R[A] \supseteq S[B] \notin Cn(I)$, then there must be a database db' which satisfies I but not $R[A] \supseteq S[B]$ by definition, thus $db' \models R[A] \not\supseteq S[B]$. Now we extend db' for each $T[C] \not\supseteq U[D] \in I'$ by one tuple, such that $db' \models I'$. This can be done by duplicating rows, and changing the mentioned values by extending the domain. By consistency, $db' models I$ remains. Thus we have $db' \models I \cup I'$ and $db' \models R[A] \not\supseteq S[B]$, which is a contradiction. Therefore, $R[A] \supseteq S[B] \in Cn(I)$. □

Another interesting point is, that there are no consequences of a set of uINIs, except the trivial ones like reflexivity.

Lemma 12 *Let I' be a set of UINIs. $R[A] \supseteq S[B] \in Cn(I')$ if and only if $R[A] \supseteq S[B] \in I'$.*

Proof: **(if)** trivial.
(only-if) We know that each database db which satisfies I' also satisfies $R[A] \not\supseteq S[B]$. We show the lemma by construction of a database which satisfies I', but not $R[A] \not\supseteq S[B]$ under the assumption that $R[A] \not\supseteq S[B] \notin I'$. A database can be constructed by adding for each $T[C] \not\supseteq U[D] \in I'$ a row to the empty database with the corresponding values, i.e. $U[D] = 2$ and $T[C] = 1$.

T	...	C	...
	1 ... 1	1	1 ... 1

U	...	D	...
	1 ... 1	2	1 ... 1

If $S[B] = U[D]$ then we use the same value as B for the attribute A in the corresponding row. thus, we get a database db' with $db' \models I'$ and $db' \not\models R[A] \not\supseteq S[B]$. Therefore, $R[A] \not\supseteq S[B] \in I'$. □

The previous lemmas clarifies the following inference rule which state that only UINIs can follow from a set of UINDs and UINIs:

Definition 12 (Inference of UINIs) *Inference rules for UINIs are given by:*

$$UI1 : \quad \frac{R[A] \supseteq S[B],\ R[A] \not\supseteq T[C]}{S[B] \not\supseteq T[C]}$$

$$UI2 : \quad \frac{R[A] \supseteq T[C],\ S[B] \not\supseteq T[C]}{S[B] \not\supseteq R[A]}$$

Theorem 7 *The rules are sound.*

Proof: Can easily be seen by contraposition of U2 □

function Infer-UINI($A \not\supseteq B, I, I'$) : *bool*;
begin
$\Delta := Closure(B, I)$
$\overline{\Delta} := \overline{Closure}(A, I)$
for each $C \in \Delta$ **do**
  for each $D \in \Delta$ **do**
    if $D \not\supseteq C \in I'$ **then** succeed
end

Table 4: Membership Algorithm for UINIs

Theorem 8 *Let I be a set of UINDs, and I' a set of UINIs.*
If $I \cup I$ is consistent then $I \cup I' \cup \{R[A] \supseteq S[B]\}$ is inconsistent iff $A \not\supseteq B \in Cn(I \cup I')$.

Proof: (if) trivial
(only-if) By lemma it suffices to prove that $T[C] \supseteq U[D] \in Cn(I \cup \{R[A] \supseteq S[B]\})$ for some $T[C] \not\supseteq U[D] \in I'$. It is shown that $I \cup \{T[C] \not\supseteq U[D]\} \vdash R[A] \not\supseteq S[B]$, whereas $\vdash$ is the inference in respect of the inference rules of I and I'. Assume that $I \cup I'$ is consistent. This implies $A \supseteq B \notin Cn(I)$ (see Lemma). By completeness of the UIND axiomatization it is known $I \not\vdash R[A] \supseteq S[B]$. Suppose without loss of generality that $R[A] \neq S[B]$ then the transitivity rule is the only one, which we can use to infer from $I \cup \{R[A] \supseteq S[B]\}$ the UIND $T[C] \supseteq U[D]$. This can be done with a chain in I : $\{T[C] \supseteq V[X_1], V[X_1] \supseteq V'[X_2], \ldots, V''[X_i] \supseteq R[A], S[B] \supseteq V'''[X_{i+1}], V''''[X_n] \supseteq T[D]\}$. This implies: $I \vdash T[C] \supseteq R[A], S[B] \supseteq U[D]$. Now it can be inferred with NU1 from $T[C] \supseteq R[A], T[C] \not\supseteq U[D]$ that $R[A] \not\supseteq S[D]$ and from this with NU2 and $S[B] \not\supseteq T[D]$ we can infer $R[A] \not\supseteq S[B]$. □

Corollary 4 *The formal system of UINDs and UINIs, given by the inference rules of definition 10 and 12, is complete.*

6 Inference of UINIs

In this section we discuss we investigate the inference of unary inclusion dependencies. Therefore, we adapt the membership problem to UINDs and UINIs.

The algorithm, which is listed in table 4, for the membership problem is mainly based on the observation, that only one UINI participate in the inference process of an UINI. Therefore, the main work is done by computing the closure and the reverse closure of the given attributes. Also, we have to recognize cycles in the closure. We state correctness and completeness by the following proposition:

Proposition 1 *Let $R[A]$, $S[B]$, $T[C]$ and $U[D]$ be attributes of a database, I and I' be sets of UINDs, respectively UINIs.*

$R[A] \not\supseteq S[B] \in Cn(I \cup I')$ if and only if there is a $T[C] \not\supseteq U[D] \in I'$, $T[C] \in Closure(R[A], I)$, and $U[D] \in \overline{Closure}(S[B], I)$.

Proof: (if) can be seen by the corollary above.
(only-if) By lemma 12 it is obvious that if $R[A] \not\supseteq S[B] \in Cn(I \cup I')$, then there is a $T[C] \not\supseteq U[D] \in I'$ and $R[A] \not\supseteq S[B] \in Cn(I \cup \{\not\supseteq U[D]\})$. By the corollary, the proposition can immediately seen. □

It takes $O(n^2)$ time for closure and $\overline{Closure}$ with n attributes in the database, because there can be $O(n^2)$ dependencies. Including recognizing cycles it takes $O(n^4)$. Combination for attributes can be $O(n^2)$.

7 Interaction of FDs and UINDs

Now the combination of FDs and UINDs is discussed. We simplify the notation of UINDs by not mentioning any more the relation of the UINDs, but it is easy to see how to expand the UINDs to get the right notation.

There is no interaction between standard FDs and UINDs in the unrestricted case. There are interactions in the finite case, but in general there are infinitely many axioms. They can be described by a set of inference rules, [Kanellakis, 1990].

$$CD: \quad \frac{A_0 \rightarrow A_1, A_1 \supseteq A_2, \ldots, A_{k-1} \rightarrow A_k, A_k \supseteq A_0}{A_1 \rightarrow A_0, A_2 \supseteq A_1, \ldots, A_k \rightarrow A_{k-1}, A_0 \supseteq A_k}$$

for each odd positive integer k.

The next question is: Is there any interaction between the dependencies and their counterparts in the finite case.

In the first example it is the case that $A \supseteq D$ and $D \not\supseteq A$. This means, that in A are really more values than in B. Therefore D can not determine A: $D \not\rightarrow A$. So there exists at least one relationship between UINDs, UINIs and FIs.

Another inference rule is given by the observation that if the FD $A \rightarrow B$ is valid and if $B \not\rightarrow A$ then it must be the case that each value of A cannot appear in B, because the same values in A have the same values in B and at least for two same values in B we have two different values in A. This means $B \not\supseteq A$.

A general inference rule is proposed, whereas for each odd positive integer k there is one inference rule:

$$CDI: \quad \frac{A_0 \rightarrow A_1, A_1 \supseteq A_2, \ldots, A_{k-1} \rightarrow A_k, A_k \supseteq A_0, A_{i+1} o' A_i}{A_l o A_{l+1}}$$

whereas for one l there is a missing element in the chain, $i, l \leq k$ and if l is even, then $o = \not\rightarrow$ else $o = \not\supseteq$ and if i is even then $o' = \not\rightarrow$ else $o' = \not\supseteq$.

Lemma 13 *CDI is sound.*

Proof: To see soundness of this inference rule it have to be remarked that both independencies express something about the cardinality of the values. If A has more values than B, then it is the case that the FI $B \not\rightarrow A$ and the UINI $B \not\supseteq A$ holds. It becomes clear with contraposition that if $A_0 \rightarrow A_1$ holds, this means, that $card(A_0) \geq card(A_1)$ and if $A_1 \supseteq A_2$ holds this means also $card(A_1) \geq card(A_2)$. This means that $card(A_{l+1}) \geq card(A_l)$. If $i = l$ and l is even it is the case that $card(A_l) \not\geq card(A_{l+1})$. This means $card(A_{l+1}) > card(A_l)$ and therefore $A_l \not\rightarrow A_{l+1}$. If $i \neq l$ then there is the case that $card(A_i) \geq card(A_{i+1})$ and $card(A_{i+1}) \not\geq card(A_i)$. It follows that $card(A_i) > card(A_{i+1})$ and that $card(A_{l+1}) > card(A_l)$. This means that for the corresponding o it is the case that $A_l \not\rightarrow A_{l+1}$ or $A_l \not\supseteq A_{l+1}$. □

The sets of FDs and UINDs are partitioned to a new set called DEP and the FIs and UINIs to a set called INDEP for proving completeness. We show how the elements of DEP and $INDEP$ interact.

Lemma 14 *Let DEP and INDEP be sets of dependencies and independencies respectively, and let the sets be consistent. $A_i \not{o} A_l \in Cn(INDEP \cup DEP)$ if and only if $A_i \not{o} A_l \in Cn(DEP)$ with $o = \rightarrow$ or $o = \supseteq$.*

Proof: (if) trivial.
(only-if) Case 1) $o = \rightarrow$: This means that $A \rightarrow B \in Cn(DEP \cup INDEP)$. Assume that $A \rightarrow B \notin Cn(DEP)$, then there must be a database db with $db \models DEP$ and $db \not\models A \rightarrow B$. db can be expanded by new values that each independency of INDEP is satisfied and the dependencies of DEP remains valid by consistency. But this means the case that $db \models DEP \cup INDEP$ and $db \not\models A \rightarrow B$ which is a contradiction. Therefore $A \rightarrow B \in Cn(DEP)$.
Case 2 analogously to case 1 □

Next, it is shown that elements of INDEP do not work together.

Lemma 15 *Let INDEP be a set of independencies. $A \not{o} B \in Cn(INDEP)$ if and only if $A \not{o} B \in INDEP$ with $o = \rightarrow$ or $o = \supseteq$.*

Proof: (if) trivial.
(only-if, sketch) Assume that $A \not{o} B \in Cn(INDEP)$ and $A \not{o} B \notin INDEP$. As above we can construct a database which satisfies each elements of $INDEP$ but no other. □

The next lemma shows the completeness of the inference rules.

Lemma 16 *If DEP and INDEP are consistent then $DEP \cup INDEP \models A_l \not{o} A_{l+1}$ iff $DEP \cup INDEP \cup \{A_l\ o\ A_{l+1}\}$ is inconsistent whereas o stands for $\rightarrow$ or $\supseteq$ and A_l and A_{l+1} are some attributes.*

Proof: (only-if) is trivial.
(if) By lemma the assumption can be transformed to $A_{i+1} o' A_i \in Cn(DEP \cup \{A_l o A_{l+1}\})$ for some $A_{i+1} \not{o}' A_i \in INDEP$, whereas A_{i+1} and A_i are some attributes. A stronger result is shown that this implies $DEP \cup \{A_{i+1} \not{o}'\ A_i\} \vdash A_l \not{o}\ A_{l+1}$. Assume $DEP \cup INDEP$ is consistent. This implies $A_l o A_{l+1} \notin Cn(DEP)$. There are four cases: (We denote with Σ the set of FDs.)

1. $A_{i+1} \rightarrow A_i \in Cn(DEP\ \cup\ \{A_l \rightarrow A_{l+1}\})$
2. $A_{i+1} \supseteq A_i \in Cn(DEP\ \cup\ \{A_l \rightarrow A_{l+1}\})$
3. $A_{i+1} \rightarrow A_i \in Cn(DEP\ \cup\ \{A_l \supseteq A_{l+1}\})$

4. $A_{i+1} \supseteq A_i \in Cn(DEP \ \cup \ \{A_l \supseteq A_{l+1}\})$

ad 1: case 1: If $A_{i+1} \to A_i \in Cn(\Sigma \cup \{A_l \to A_{l+1}\})$, then it is known by completeness of FDs and FIs that $DEP \cup \{A_{i+1} \to A_i\} \vdash A_l \to A_{l+1}$.
case 2: If $A_{i+1} \to A_i \notin Cn(\Sigma \cup \{A_l \to A_{l+1}\})$, then it is known by completeness of DEP, that $DEP \ \cup \ \{A_l \to A_{l+1}\} \vdash A_{i+1} \to A_i$ and this means the inference rule CD can be applied and therefore $A_0 \to A_1, \ldots, A_l o A_{l+1}, \ldots, A_k \supseteq A_0 \in Cn(DEP)$. By I1 it can be inferred $DEP \cup \{A_{i+1} \to A_i\} \vdash A_l \not\to A_{l+1}$.
ad 2: Assume that $A_0 \to A_1, \ldots, A_{l-1} \supseteq A_l, A_{l+1} \supseteq A_{l+2} \ldots, A_k \supseteq A_0 \in Cn(DEP)$. Therefore it can be inferred from $DEP \vdash A_0 \to A_1, \ldots, A_{l-1} \supseteq A_l, A_{l+1} \supseteq A_{l+2} \ldots, A_k \supseteq A_0$ and from $DEP \cup \{A_{i+1} \not\supseteq A_i\} \vdash A_l \not\to A_{l+1}$.
ad 3 and ad 4 are analog to ad1 and ad2. □

Corollary 5 *The inference of FDs, FIs, UINDs and UINIs, given by the axioms and rules of definition 2, 3, 10, 12, CD and CDI, is sound and complete.*

It is shown by Kleene that each axiomatization can be transformed into an finite axiomatization by the introduction of new predicates, i.e. in order to avoid infinite many inference rules. Here the language is expanded with a new kind of dependencies, the *cardinality dependencies* like in [Kanellakis et al., 1983].

Definition 13 (Cardinality Dependencies) *The cardinality dependency $|A| \geq |B|$ is true iff the cardinality of the attribute A is greater or equal the cardinality of the attribute B. Strictly greater is abbreviated with $|A| > |B|$.*

In [Kanellakis et al., 1983] a sound and complete axiomatization of functional and inclusion dependencies regarding cardinality dependencies is given by the following axiom and rules:

Definition 14 (Finite Axiomatization of FDs and UINDs) *An axiomatization is given by:*

- *The rules FD1, FD2, FD3 and U1 , U2.*
- N1*: reflexivity axiom, $|A| \geq |A|$.*
- N2*: transitivity rule, $\frac{|A| \geq |B|, |B| \geq |C|}{|A| \geq |C|}$.*
- N3*: $\frac{A \to B}{|A| \geq |B|}$*
- N4*: $\frac{A \supseteq B}{|A| \geq |B|}$*
- N5*: $\frac{A \to B, |B| \geq |A|}{B \to A}$*
- N6*: $\frac{A \supseteq B, |B| \geq |A|}{B \supseteq A}$*
- N7*: $\frac{R{:}\emptyset \to A, S{:}B}{|S[B]| \geq |R[A]|}$*

N7 is necessary if nonstandard dependencies are allowed and interrelational dependencies are taken into consideration. The infinite set of inference rules is replaced by the following rules regarding cardinality dependencies.

Definition 15 (Finite Axiomatization with Cardinalities) *An axiomatization is given by:*

- *The rules FI1, FI2, FI3 and NU1, NU2 and of definition 8.*
- I1: $\frac{|A|\geq|B|,\ A\not\twoheadrightarrow B}{B\not\twoheadrightarrow A}$
- I2: $\frac{|A|\geq|B|,\ A\not\supseteq B}{B\not\supseteq A}$
- I3: $\frac{|A|>|B|}{B\not\twoheadrightarrow A}$
- I4: $\frac{|A|>|B|}{B\not\supseteq A}$
- I5: $\frac{|A|>|B|}{|A|\geq|B|}$
- I6: $\frac{A\supseteq B, B\not\supseteq A}{|A|>|B|}$
- I7: $\frac{A\rightarrow B, B\not\twoheadrightarrow A}{|A|>|B|}$
- I8: $\frac{|A|>|B|, |B|\geq|C|}{|A|>|C|}$
- I9: $\frac{|A|\geq|B|, |B|>|C|}{|A|>|C|}$

Lemma 17 (Equivalence) *If AoB is inferred by the inference rules of definition 14 then AoB can be inferred by the rules of definition 15.*

Proof: Assume $\{A_{i+1} \not\twoheadrightarrow A_i, A_0 \rightarrow A_1, \ldots, A_{l-1} \rightarrow A_{l+1} \rightarrow A_{l+2}, \ldots, A_k \supseteq A_0\}$ with $l, i \leq k$. First consider the case $l \neq i$. Then it can be inferred by $I7$ from $A_i \rightarrow A_{i+1}$ and $A_{i+1} \not\twoheadrightarrow A_i$ that $|A_i| > |A_{i+1}|$. By using rules $N3$, $N4$, $I9$ and $I8$ that $|A_{l+1}| > |A_i|$. If l is even, it is known by rule $I3$ that $A_i \not\twoheadrightarrow A_{l+1}$ and if l is odd, it is known $A_i \not\supseteq A_{l+1}$. For $l = i$ it is known by $N3$, $N4$ and $N2$ that $|A_{i+1}| \geq |A_i|$. Then by $I1$ it is known $A_i \not\twoheadrightarrow A_{i+1}$. The other cases are analog. □

FDs and UINDs are recursive, therefore their counterparts are decidable. This implies a decision procedure, which can be expressed as a formal system.

Corollary 6 *If $DEP \cup INDEP \models o$, then there is a formal system which is sound and complete to decide if $DEP \cup INDEP \vdash AoB$ by the rules of definition 15.*

8 Complexity of Independency Inference

We regard in this section the membership problem in respect to unary FDs and UINDs. The test of FDs and UINDs alone takes $O(n)$ (cf. [Kanellakis, 1990]) regarding n as the number of attributes. In the finite case, the test regarding combination of FDs and UINDs takes $O(n^3)$ and $O(n)$ if we only regard unary FDs.

We assume that the cardinalities of the attributes are all known. They can be represented by a list with a total order. We change therefore the previous algorithms to algorithm listed in tabel 5, but the costs remains mainly the same. Assume that the number of attributes is n. If $o = \rightarrow$ then we need n times the time of infer-FI time which is $O(n^3)$, and otherwise we need the same with the appropriate data structures.

9 Multivalued Independencies

A strong relationship exists between FIs and multivalued dependencies (MVD), because they state that some attributes are **independent** from others attributes. Here we present in a rough sketch results about multivalued independencies which can be achieved by the already used techniques. Thus we left out the proofs.

A definition of multivalued dependencies is given by:

```
function Infer–FI–UINI(A o/ B, Card, Σ, Σ', I, I') : bool;
begin
if o =→ then
        begin
        if  B > A then succeed
           else if B ≥ A and Infer–FI(B ↛ A, I, I') then succeed
                  else if Infer–FI(A ↛ B, I, I') then succeed
        end
        else
        if  B > A then succeed
           else if B ≥ A and Infer–UINI(B ⊉ A, I, I') then succeed
                  else if Infer–UINI(A ⊉ B, I, I') then succeed
end
```

Table 5: Membership Algorithm for FIs and UINIs

Definition 16 (Multivalued Dependency (MVD)) *The MVD* $X \twoheadrightarrow Y$ *holds in r iff for all tuples* t_1 *and* t_2 *in r, if* $t_1[X] = t_2[X]$, *then there are tuples* t_3 *and* t_4 *in r such that*

(i) $t_3[X] = t_1[X], \quad t_3[Y] = t_1[Y] \quad and \quad t_3[Z] = t_2[Z]$

(ii) $t_4[X] = t_2[X], \quad t_4[Y] = t_2[Y] \quad and \quad t_4[Z] = t_1[Z]$

A formal system is given by:

Definition 17 (Axiomatization of MVDs) X, Y *and* Z *are sets of attributes, with* U *is the set of all attributes. An axiomatization of MVDs is given by:*

$MDc : (Complementation)$ *If* U *is the universe of attributes, then* $\models X \twoheadrightarrow U - X$

$MDa : (Augmentation)$ $\frac{X \twoheadrightarrow Y}{XZ \twoheadrightarrow YZ}$

$MDd : (Difference)$ *if* $Y \cap Z = \{\}$ *then* $\frac{X \twoheadrightarrow Y, Z \twoheadrightarrow Y_1}{X \twoheadrightarrow Y - Y_1}$

This axiomatization does again constitute an inference relation $\vdash$. The counterpart to this dependencies are the so called multivalued independencies, which can be defined as:

Definition 18 (Multivalued Independency (MVI)) *The MVI* $X \not\twoheadrightarrow Y$ *holds in r if there are tuples* t_1, t_2 *in r, if* $t_1[X] = t_2[X], t_1[Y] \neq t_2[Y], t_1[Z] \neq t_2[Z]$ *and is no tuple* t_3 *with Z = U - X - Y:*

(i) $t_3[X] = t_1[X], \quad t_3[Y] = t_1[Y] \quad and \quad t_3[Z] = t_2[Z]$

The definition 18 shows, that MVDs and MVIs do not work together in the sense, that there are no inference rules, to infer from MVIs any MVDs. Also MVIs do not interact together.

The following inference rules are proposed, which constitute an inference relation $\vdash_m$. The soundness follows by contraposition of the axiomatization of MVDs. The completeness can be seen by similar arguments.

Definition 19 (Inference of MVIs) *Inference rules for the MVIs are given by with U as the universe of attributes:*

$$MIr: \quad \frac{XZ \not\twoheadrightarrow YZ}{X \not\twoheadrightarrow Y}$$

$$MI1: \quad if\, Y \cap Z = \{\} \; then \; \frac{X \twoheadrightarrow Y, X \not\twoheadrightarrow Y - Y_1}{Z \not\twoheadrightarrow Y_1}$$

$$MI2: if\, Y \cap Z = \{\} \; then \; \frac{Z \twoheadrightarrow Y_1, X \not\twoheadrightarrow Y - Y_1}{X \not\twoheadrightarrow Y}$$

The next step is to investigate the combination of functional and multivalued dependencies. In [Kanellakis, 1990] a formal system is given by:

Definition 20 (Interaction of FDs and MVDs) *Inference rules for the interaction between FDs and MVDs are given by:*

$$MFDt: \quad \frac{X \rightarrow Y}{X \twoheadrightarrow Y}$$

$$MFDi: \quad if\ Y \ \cap Z = \{\} \; then \; \frac{X \twoheadrightarrow Y, Z \rightarrow Y_1}{X \rightarrow Y \cap Y_1}$$

The following rules reflect the interaction between functional and multivalued independencies:

Definition 21 (Interaction of FIs and MVIs) *Inference rules for the interaction between FIs and MVIs are given by:*

$$MFI1: \quad \frac{X \not\twoheadrightarrow Y}{X \not\rightarrow Y}$$

$$MFI2: \quad if\ Y \cap Z = \{\} \; then \; \frac{X \twoheadrightarrow Y, X \not\rightarrow Y \cap Y_1}{Z \not\rightarrow Y_1}$$

$$MFI3: \quad if\ Y \cap Z = \{\} \; then \; \frac{Z \rightarrow Y_1, X \not\rightarrow Y \cap Y_1}{X \not\twoheadrightarrow Y}$$

10 Related Work

Query optimization can be regarded as the process of transforming a query Q into another query Q' that can be evaluated more efficiently, as mentioned by Chakravarthy et al. [Chakravarthy et al., 1990]. Semantic query optimization (SQO) is mainly based on the use of semantic knowledge during the optimization process. Thus, the user is motivated to concentrate on the application rather than forming queries with explicit semantic knowledge of the application.

So far, the main problem is to provide the optimizers with semantic knowledge about the database during SQO. Obviously, the only kind of semantic knowledge which is always available in relational databases management systems (DBMS) are integrity constraints like primary or foreign keys. Thus, Chakravarthy et al. [Chakravarthy et al., 1990] have defined SQO in respect of integrity constraints as to transform a query into one which is semantically equivalent to the original query, but which can be executed more efficiently. King [King, 1981] argued that the costs of evaluating the transformed query plus the transformation costs should be lower than the costs of evaluating the original query. Semantic equivalence means that the transformed query has the same answer as the original query on all database states satisfying the integrity constraints. Jarke et al. [Jarke et al., 1984] have shown several ways to use functional dependencies for SQO. But the constraints provided by a DBMS are few and often too general in the sense that they are valid in all possible database states.

Another way is to provide SQO with semantic knowledge by hand which also seems no adequate technique. For example, King [King, 1981] uses constraints on attribute values to optimize queries by his system QUIST. Zhang and Ozsoyoglu [Zhang and Ozsoyoglu, 1994] have presented also techniques for semantic query optimization which are based on implication constraints and referential constraints. Implication constraints are a generalization of functional dependencies and referential constraints of inclusion dependencies.

The arise of knowledge discovery in databases (KDD) offers a new approach to solve both problems: provides SQO automatically with constraints and extends them to constraints which precisely reflects the present content of the database. Siegel has reported this by the first time [Siegel, 1988] and [Siegel et al., 1991]. Such constraints have been termed, for example, *Database Abstractions* in [Hsu and Knoblock, 1993], *Metadata* in [Siegel and Madnick, 1991], and *Meta Knowledge* in [Schlimmer, 1991]. Also, Hsu and Knoblock [Hsu and Knoblock, 1993] have shown the benefits of optimization techniques based on automatically discovered constraints. But we have to keep in mind that these constraints are only valid in the present state of the database and therefore describe the content of the database precisely. The constraints may become invalid, if the database changes. Therefore, we have to maintain the discovered knowledge, if we use it more than once.

Problems arise if knowledge discovery is applied to real world databases which are continuously in use and large. Therefore, knowledge discovery in databases is only allowed to take a small portion of the system resources and the use of independencies is one way to improve this process.

But FIs can also be used directly for semantic query optimization in order to refute or to simplify queries. For example, the already mentioned approach of Zhang and Ozsoyoglu [Zhang and Ozsoyoglu, 1994] use implication and referential constraints which are generalizations of functional and inclusion dependencies.

Gottlob and Libkin [Gottlob and Libkin, 1990] have shown that the MAX-set, introduced by Mannila and Räihä [Mannila and Raiha, 1986], can be written and interpreted as functional independencies. But in both works a closed world is assumed, which makes the concept of FIs superfluous, because FIs are here an alternative way of representing FDs. In contrast to this, we are not forced to know all FDs or FIs. Our system still works if we have only proper subsets of FDs and FIs in order to prevent the worst cases of exponentially many FDs in a relation. This makes their approaches to ours absolutely incomparable.

Comparable to our approach in order to discover functional dependencies, there are similar's by Mannila and Räihä [Mannila and Räihä, 1991], Schlimmer [Schlimmer, 1993], Savnik and Flach [Savnik and Flach, 1993], and Dehaspe et al. [Dehaspe et al., 1994]. Mannila and Räihä have investigated the problem of inferring FDs from example relations in order to determine database schemes. But they do not use a complete inference relation regarding independencies. Savnik and Flach have investigated a special data structure for the FDs. Briefly, they start with a bottom–up analysis of the tuples and construct a *negative cover*, which is a set of FIs. In the next step they use a top–down search approach. They check the validity of a dependency by searching for FIs in the negative cover. Also, the negative cover is not complete regarding a classical consequence relation. Schlimmer also uses a top–down approach, but in conjunction with a hash–function in order to avoid redundant computations. However, he does not use a complete inference relation even regarding functional dependencies. Also do Dehaspe et al. because their inferences are based on Θ–subsumption. In addition, the verification is based on theorem proving which is not suitable for real world databases.

In general, these authors do not use a relational database like OracleV7 or any other commercial DBMS. In such case, we argue that the proposed algorithm and approaches have to redesign according the set oriented interface of a relational database system. For example, the concept of the negative cover has only advantages if the tuples can be accessed directly, i.e. the tuples are stored in the main memory as Prolog–facts. Savnik and Flach have introduced it because the complexity for testing contradiction of the FDs is reduced.

In contrast to these approaches our purpose is different because we maintain the discovered FDs in order to use them all the time by semantic query optimization.

In addition, we argue that by using a relational database system, the higher complexity of the complete inference relation is justified by the size of a real world database.

Thus, our approach has two advantages: it does not presumes the closed world assumption, which does not make sense in a knowledge discovery or machine learning environment, cf. [Bell and Weber, 1993]. The second advantage is, that this approach guarantees minimal database access by completeness of the axiomatizations.

11 Conclusions

We have presented a more detailed view of the implication or membership problem $\sigma \in Cn(\Sigma)$. We use the concept and the axiomatization of independencies in our system twofold: First, independencies help to minimize the number of accesses to the database w.r.t. the discovery of dependencies and their maintenance, because the alternative to a complete inference would be a more or less exhaustive test of FDs on the database. Usually, real world databases are very large, the number of tuples is much larger than the number of attributes. Thus, the main costs of database management systems are caused by reading from secondary memory. Therefore, a single saved database query makes up for the costs of inferring FDs and FIs. This is true for the maintenance, too.

Second, independencies can directly be used for semantic query optimization in order to refute or simplify queries.

Currently, we integrate these discovered and maintained constraints in a semantic query optimizer.

References

[Beeri et al., 1984] Beeri, C., Dowd, M., Fagin, R., and Statman, R. (1984). On the structure of armstrong relations for functional dependencies. *Journal of the ACM*, 31(1):30–46.

[Bell and Brockhausen, 1995] Bell, S. and Brockhausen, P. (1995). Discovery of constraints and data dependencies in databases (extended abstract). In Lavrac, N. and Wrobel, S., editors, *Machine Learning: ECML-95 (Proc. European Conf. on Machine Learning, 1995)*, Lecture Notes in Artificial Intelligence 914, pages 267 – 270, Berlin, Heidelberg, New York. Springer Verlag.

[Bell and Weber, 1993] Bell, S. and Weber, S. (1993). A three-valued logic for inductive logic programming. Technical Report 4, Dortmund University, Computer Science VIII, 44221 Dortmund Germany.

[Chakravarthy et al., 1990] Chakravarthy, U. S., Grant, J., and Minker, J. (1990). Logic–based approach to semantic query optimization. *ACM Transaction on Database Systems*, 15(2).

[Dehaspe et al., 1994] Dehaspe, L., Laer, W. V., and Raedt, L. D. (1994). Applications of a logical discovery engine. In Wrobel, S., editor, *Proc. of the Fourth International Workshop on Inductive Logic Programming*, pages 291–304, St. Augustin, Germany. GMD.

[Gärdenfors, 1988] Gärdenfors, P. (1988). *Knowledge in Flux — Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, MA.

[Gottlob and Libkin, 1990] Gottlob, G. and Libkin, L. (1990). Investigations on armstrong relations, dependency inference, and excluded functional dependencies. *Acta Cybernetica*, 9(4).

[Hsu and Knoblock, 1993] Hsu, C.-N. and Knoblock, C. A. (1993). Learning database abstractions for query reformulation. In *Knowledge Discovery in Database, Workshop, AAAI-93*.

[Janas, 1988] Janas, J. M. (1988). Covers for functional independencies. In *Conference of Database Theory*. Springer, Lecture Notes in Computer Science 338.

[Jarke et al., 1984] Jarke, M., Clifford, J., and Vassiliou, Y. (1984). An optimizing prolog front–end to a relational query system. *ACM SIGMOD*.

[Kanellakis, 1990] Kanellakis, P. (1990). *Formal Models and Semantics, Handbook of Theoretical Computer Science*, chapter Elements of Relational Database Theory, 12, pages 1074 – 1156. Elsevier.

[Kanellakis et al., 1983] Kanellakis, P., Cosmadakis, S., and Vardi, M. (1983). Unary inclusion dependencies have polynomial time inference problems. *Proc. 15th Annual ACM Symposium on Theory of Computation*.

[King, 1981] King, J. J. (1981). Query optimization by semantic reasoning. Technical Report STAN-CS-81-857, Stanford University.

[Maier, 1980] Maier, D. (1980). Minimum covers in the relational database model. *Journal of the ACM*, 27(4):664 – 674.

[Mannila and Raiha, 1986] Mannila, H. and Raiha, K.-J. (1986). Design by example: An application of armstrong relations. *Journal of Computer and System Science*, 33.

[Mannila and Räihä, 1991] Mannila, H. and Räihä, K.-J. (1991). *The design of relational databases*. Addison-Wesley.

[Paredaens et al., 1989] Paredaens, J., de Bra, P., Gyssens, M., and van Gucht, D. (1989). *The Structure of the Relational Database Model*. Springer Verlag Berlin Heidelberg.

[Savnik and Flach, 1993] Savnik, I. and Flach, P. (1993). Bottum-up indution of functional dependencies from relations. In Piatetsky-Shapiro, G., editor, *KDD-93: Workshop on Knowledge Discovery in Databases*. AAAI.

[Schlimmer, 1993] Schlimmer, J. (1993). Using learned dependencies to automatically construct sufficient and sensible editing views. In Piatetsky-Shapiro, G., editor, *KDD-93: Workshop on Knowledge Discovery in Databases*. AAAI.

[Schlimmer, 1991] Schlimmer, J. C. (1991). Database consistency via inductive learning. In *Eight International Conference on Machine Learning*.

[Siegel and Madnick, 1991] Siegel, M. and Madnick, S. M. (1991). A metadata approach to resolving semantic conflicts. In *Conference on Very Large Databases*.

[Siegel et al., 1991] Siegel, M., Sciore, E., and Salveter, S. (1991). Rule discovery for query optimization. In *Knowledge Discovery in Databases*, chapter 24. AAAI Press, Menlo Park.

[Siegel, 1988] Siegel, M. D. (1988). Automatic rule derivation for semantic query optimization. In *Second International Conference on Expert Database Systems*.

[Zhang and Ozsoyoglu, 1994] Zhang, X. and Ozsoyoglu, Z. M. (1994). Reasoning with implicational and referential constraints in semantic query optimization. In *Workshop on Constraints and Databases, Post–ILPS*.