The Expanded Implication Problem of Data Dependencies\textsuperscript{1}

LS–8 Report 16

Siegfried Bell

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

Dortmund, 12. June 1995

\textsuperscript{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
The Expanded Implication Problem of Data Dependencies

LS-8 Report 16

Siegfried Bell

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

Dortmund, 12. June 1995

2Parts 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 \( \Sigma \) and a dependency \( \sigma \), find whether \( \Sigma \) entails \( \sigma \), 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 \( \Sigma \) is known to be valid and a set of dependencies \( \Sigma' \) 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 \( \sigma \) already entailed by \( \Sigma \) and \( \Sigma' \)? Since there are interactions between \( \Sigma \) and \( \Sigma' \), 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:

\[
V := \{\}
\]

\[
\text{for each } s \in S \text{ do}
\]

\[
\quad \text{if } test(r, s) \text{ then } V := V \cup \{s\}
\]

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:

\[
V := \{\}
\]

\[
\text{for each } s \in S \text{ do}
\]

\[
\quad \text{if } s \notin Cn(V) \text{ then if } test(r, s) \text{ then } V := V \cup \{s\}
\]

But we still have unnecessary tests in our algorithm. For example, if we look at functional dependencies: Let be \( A \rightarrow B \in V \) and \( test(r, C \rightarrow B) \) evaluates to \( false \). Thus, we do not have to test \( C \rightarrow A \), 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:

\[
V := \{\}
\]

\[
\text{for each } s \in S \text{ do}
\]

\[
\quad \text{if } s \notin Cn(V) \text{ and } \neg s \notin Cn(V) \text{ then if } test(r, s) \text{ then } V := V \cup \{s\}
\]

\[
\quad \text{else } V := V \cup \{-s\}
\]
FUNCTIONAL INDEPENDENCIES

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>NULL</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>NULL</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 1: Equality operator $\equiv$

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 $A, B, C, ...$ 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 $\text{null}$ values because the ISO standard permits $\text{null}$ values in any attribute of a candidate key. Therefore, we adopt a special equality operator for the definition of the FDs, $\equiv$, which is illustrated in figure 1.

Now, we can define the consequences of a set of dependencies with: Let $\Sigma$ be a set of functional dependencies, then $X \rightarrow Y$ is a consequence of $\Sigma$ or $X \rightarrow Y \in CONS(\Sigma)$: whenever a relation satisfies $\Sigma$, 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 $\frac{X \rightarrow Y}{X \cup W \rightarrow Y W}$
- $FD3$ (Transitivity) $\frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z}$

Some well known rules like union and complementation are logically implied by this system. The closure of attributes $X$ regarding a set of FDs $\Sigma$ is defined as: $\text{closure}(X, \Sigma) = \{Y | X \rightarrow Y \in CONS(\Sigma)\}$ 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] a functional dependencies\(^1\) are introduced, but they are a sort of semantic constraints and much stronger than our functional independencies.

**Definition 2 (Functional Independence (FI))** \(X \not\rightarrow Y\) denotes a functional independence. 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 \(t_1[X] = t_2[X]\) and \(t_1[Y] \neq t_2[Y]\).

The consequences of FDs and FIs are defined as follows: Let \(\Sigma\) be a set of FDs and \(\Sigma'\) a set of FIs. \(Cl(\Sigma \cup \Sigma') := \{\sigma\} \text{ for each relation } r \text{ if } r \models \Sigma \cup \Sigma'\text{, then } r \models \sigma\}, \text{ where } \sigma \text{ 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 \(\Sigma\) be a set of FDs, \(\Sigma'\) a set of FIs and \(\Sigma \cup \Sigma'\) be consistent. \(X \rightarrow Y \in \mathcal{C}(\Sigma \cup \Sigma')\) if and only if \(X \rightarrow Y \in \mathcal{C}(\Sigma)\).

**Proof:** (if) is trivial by monotonicity of \(\mathcal{C}\).

(only-if) Assume that \(X \rightarrow Y \notin \mathcal{C}(\Sigma)\) and \(X \rightarrow Y \in \mathcal{C}(\Sigma \cup \Sigma')\). 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 \(t_1[X] = t_2[X]\) and \(t_1[Y] \neq t_2[Y]\). 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 \(\Sigma\), 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 \(\Sigma\) and \(\Sigma \cup \Sigma'\) by construction, but not \(X \rightarrow Y\), which is a contradiction.

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 \(\Sigma'\) be a set of FIs. If \(X \not\rightarrow Y \in \mathcal{C}(\Sigma')\), then there exists a FI \(V \not\rightarrow W \in \Sigma'\) with \(X \subseteq V\), and \(W \cap Y \neq \emptyset\).

**Proof:** Assume that \(X \not\rightarrow Y \in \mathcal{C}(\Sigma')\), and \(V \not\rightarrow W \notin \Sigma'\) with \(X \subseteq V\), and \(W \cap Y \neq \emptyset\), 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 \(X \not\rightarrow Y \notin \mathcal{C}(\Sigma')\) which is a contradiction to the assumption.

\(^1\)The definition of the AD \(X \# Y\) requires that for each tuple \(t\) there exists a tuple \(t'\) so that \(t[X] \neq t'[X]\) and \(t[Y] \neq t'[Y]\).
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 \( S_i \not\in T_i \in \Sigma' \), the following condition holds \( t_i[S_i] \neq t_{i+n}[S_i] \). 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 \( S_i \) with \( S_i \not\in T_i \in \Sigma' \) and \( t_i[T_i] \neq t_{i+n}[T_i] \). 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 \( t_i[X] \neq t_{i+n}[X] \), then also \( t_i[Y] \neq t_{i+n}[Y] \).

**Corollary 1** Let \( \Sigma \) be a set of FDs, \( \Sigma' \) 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 \( \models_{\text{Janas}} \).

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

1. \( \frac{X \not\rightarrow Y}{X \not\rightarrow Y Z} \)
2. \( \frac{XZ \not\rightarrow YZ}{XZ \not\rightarrow Y} \)
3. \( \frac{X \not\rightarrow Y Z \not\rightarrow Z}{Y \not\rightarrow X} \)

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 \( \Sigma \cup \Sigma' \not\models_{\text{Janas}} X \not\rightarrow Y \).

**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. \( \square \)

**Lemma 4** \( \{ X \rightarrow Y, Z \not\rightarrow Y \} \not\models_{\text{Janas}} Z \not\rightarrow X \).

**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 \( \Sigma \), i.e. \( \{ X \rightarrow Y \} \not\models_{\Sigma} Y \rightarrow Z \). Thus, \( Z \not\rightarrow X \) cannot be inferred. \( \square \)

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

Instead, we propose the following axiomatization:

**Definition 4 (Inference of FIs)** An inference relation \( \models_{\text{fi}} \) is given by an axiomatization of the FDs and the following inference rules:
FI1: \[
\frac{V \not\rightarrow U \cup V}{V \not\rightarrow Y}
\]

FI2: \[
\frac{X \rightarrow Y, X \not\rightarrow Z}{Y \not\rightarrow Z}
\]

FI3: \[
\frac{Y \rightarrow Z, X \not\rightarrow Z}{X \not\rightarrow Y}
\]

For example, the functional independency \(X \not\rightarrowYZ\), which is a consequence of Janas's first inference rule, can be inferred by \(\text{II}1\), as follows: we infer \(YZ \rightarrow Y\) by Armstrong's Axiom and use FI3 to infer the FI \(X \not\rightarrow YZ\) from \(X \not\rightarrow Y\) and \(YZ \rightarrow Y\). 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 \(V \not\rightarrow Y \in Cn(\{V \not\rightarrow YU\})\). This means that each relation that satisfies \(V \not\rightarrow YU\) must satisfy \(V \not\rightarrow Y\). By definition there are tuples \(t_1, t_2\) with \(t_1[V] \neq t_2[V]\) and \(t_1[YU] \neq t_2[YU]\). Since \(U \subseteq V\), it follows that \(t_1[U] \neq t_2[U]\), thus \(t_1[V] \neq t_2[V]\) and \(t_1[V] \neq t_2[V]\) and therefore \(V \not\rightarrow Y\).

- (FI2) \(\{X \rightarrow Y, X \not\rightarrow Z\}\) means there are tuples \(t_1, t_2\) with \(t_1[X] \neq t_2[X]\), \(t_1[Z] \neq t_2[Z]\) and for all tuples, particularly for \(t_1, t_2\) \(t_1[X] \neq t_2[X]\) and \(t_1[Y] \neq t_2[Y]\). Then \(t_1[Y] \neq t_2[Y]\) and \(t_1[Z] \neq t_2[Z]\) and therefore \(Y \not\rightarrow Z\).

- (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 \(\Sigma\) a set of FDs and \(X\) a set of attributes. Then the base of \(X\) is defined by:

\[
\overline{X} = \{Y \mid Y \rightarrow X \in Cn(\Sigma)\}
\]

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, \(\overline{X}\) consists of minimal sets \(S_i\) or \(\overline{X} = \{S_i \mid 1 \leq i \leq n\}\) and if \(S_j \supseteq S_i\) then \(S_j = S_i\) for \(i, j \in 1, \ldots, n\). For example, let \(\Sigma = \{AB \rightarrow C, CD \rightarrow EF\}\) be a set of FDs, then \(\overline{EF} = \{ABD, C, CD, EF\}\). 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 \(\overline{V} = \{V_j\}\) for all \(S_i\) it holds, that \(V_j \in \mathcal{P}U\setminus S_i\) and \(V_j \supseteq S_i\). 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 $\Sigma$ be a set of FDs, $\Sigma'$ a set of FIs and $\Sigma \cup \Sigma'$ consistent. We show that if $X \neq Y \in Cn(\Sigma \cup \Sigma')$ then $\Sigma \cup \Sigma' \vdash_X X \neq Y$. By corollary 1 we know that $X \neq Y \in Cn(\Sigma \cup \{R \neq S\})$ for $R \neq S \in \Sigma'$. Remember that $\overline{R}$ is the closure of $R$ w.r.t. $\Sigma$. Then we can partition the attributes of each relation which satisfy $\Sigma \cup \{R \neq S\}$ 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 $X \not\in \overline{R}$: Then we can easily construct a relation $r$ with two rows which satisfies $\Sigma \cup \{R \neq S\}$ but not $X \neq Y$ by assigning each attribute of $U$ different values. If the left hand sides of the FDs in $\Sigma$ 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 $t_1[X] \neq t_2[X]$. Thus, $r \not\models \Sigma \cup \{R \neq S\}$, but $r \models X \neq Y$. Therefore $X \subseteq \overline{R}$.

\[
\begin{array}{c|c}
U & \overline{R} \\
0... & 0...
\end{array}
\]

- Let $X \subseteq \overline{R}$, $S = S_i$ 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 $YW \rightarrow S$. By repeated application of $FI2$ we know that $\overline{R} \neq S$, and by $FI3$ that $\overline{R} \neq YW$. There are two case:

   (a) $W \subseteq \overline{R}$: Thus, by $FI1$ $\overline{R} \neq Y$ is inferred and by $FI2$ $X \neq Y$.

   (b) $W \not\subseteq \overline{R}$: Thus, we infer by $FI1$ the FI $YW' \rightarrow S$ with the attributes $W' = W \setminus \overline{R}$ and construct a counterrelation $r$ by

   \[
   \begin{array}{ccc|c}
   \overline{R} & W' & S & U \setminus S \cup W \\
   0... & 0... & 0... & 0...
   \end{array}
   \]

   Then it is easy to see that $r \models \Sigma \cup \{R \neq S\}$, but $r \not\models X \neq Y$, 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 $S_i$ which occur in a $S_i$ and the remaining attributes $U$:

   \[
   \begin{array}{ccc|c}
   \overline{R} & S \setminus \overline{R} & U \setminus S \\
   0... & 0... & 0...
   \end{array}
   \]

   Then it is easy to see that $r \models \Sigma \cup \{R \neq S\}$ but $r \not\models X \neq Y$, which is a contradiction. Thus, $Z \neq \{\}$.

3. $Z = S_i$: By assumption, we know that $Y \rightarrow S$ and $R \neq S$. By rule $FI3$ we conclude that $R \neq Y$. We know also, that $X \subseteq \overline{R}$, respectively, that $R 
\rightarrow X$. Thus, by $FI2$ we conclude that $X \neq Y$. 


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

\[ \Box \]

**Corollary 3 (Completeness and Soundness)** Let $\Sigma$ and $\Sigma'$ be a set of FDs and FIs respectively, and $\Sigma \cup \Sigma'$ be consistent. $X \neq Y \in Cn(\Sigma \cup \Sigma')$ if and only if $\Sigma \cup \Sigma' \vdash_{fi} X \neq Y$.

### 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 $\Sigma$ 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 $\Sigma \cup \Sigma' \vdash_{fi} X \neq Y$
begin
  for each $V \neq W \in \Sigma'$ do
    for each $Z \subseteq W \cap V$
      if $\Sigma \vdash_{A} V \rightarrow X$ and $\Sigma \vdash_{A} Y \rightarrow W \setminus Z$
      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 $S_i \neq T_i \in \Sigma'$ 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 \( \vdash_{f_i} \):

**Definition 6 (Minimal Cover)** Let \( \Sigma \) be a minimal set of FDs.

\( \Sigma' \) is a set of FIs and is called minimal if for all \( V \neq W \in Cn(\Sigma \cup \Sigma') \) there exists no
\( X \neq Y \in \Sigma' \) with \( \Sigma \cup \Sigma' \setminus \{X \neq Y\} \vdash_{f_i} V \neq W \).

Therefore, minimizing can be done by repeated application of \( \vdash_{f_i} \) and takes \( O(n^2) \)
time for some set of FDs and FIs.

**3.3 Initialization**

We initialize our data structure with information about primary keys and sufficient condi-
tions 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 un-
ary 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 :

\[
\begin{align*}
X \rightarrow Y & \quad \text{if } |X| \geq |Y| \\
X \rightarrow Y, |Y| \geq |X| & \quad \text{if } Y \rightarrow X
\end{align*}
\]
Theorem 3 (Soundness of $\vdash_{fc}$) The inference rules of definition 7 are sound.

**Proof:** (soundness) trivial

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 $\Sigma$ and $\Gamma$ be a set of FDs and $\Gamma$ 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 $\cap$ 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 $\cap$ is idempotent.

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 $\Sigma$ be a set of FDs and $\Gamma$ 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 $\Sigma$, but does not satisfy neither $X \rightarrow Y$ nor $Y \rightarrow X$. Again $X$ 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 $\vdash_{fc}$. 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.

<table>
<thead>
<tr>
<th>$U$</th>
<th>$X$</th>
<th>$\overline{Y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...</td>
<td>0...</td>
<td>0...</td>
</tr>
<tr>
<td>...</td>
<td>0...</td>
<td>1...</td>
</tr>
<tr>
<td>1...</td>
<td>1...</td>
<td>1...</td>
</tr>
</tbody>
</table>

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

**Theorem 4 (Completeness of $\vdash_{fc}$)** Let $\Sigma$ be a set of FDs and $\Gamma$ 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.
Theorem 5 Definition 8 establishes an inference relation $\vdash_1$ which is sound and complete.

Proof: (Soundness) is trivial.

Let $\Sigma, \Sigma'$ and $\Gamma$ 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 $\Sigma \cup \Sigma' \cup \Gamma \vdash_1 X \not\rightarrow Y$. 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
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 \( Y \not\subset X \in \text{Cn}(\Sigma \cup \Sigma' \cup \Gamma) \). Then we can infer with the first inference rule that \( \Sigma \cup \Sigma' \cup \Gamma \vdash X \not\subset Y \). Third, if \( |X| > |Y| \), then we can infer \( Y \not\subset X \) and also with the second inference rule \( X \not\subset Y \).

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. 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 \( \Sigma' \) 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

\[
\begin{array}{cccccccc}
A_1 & A_2 & A_3 & A_4 & \cdots & A_{2m-1} & A_{2m} & A_{2m+1} \\
0 & 1 & 0 & 2^{2m} + 1 & \cdots & 0 & 2(m-1)2^m & 0 \\
0 & 2 & 0 & 2^{2m} + 2 & \cdots & 0 & 2(m-1)2^m + 1 & 1 \\
3 & 0 & 0 & 2^{2m} + 3 & \cdots & 0 & 2(m-1)2^m + 2 & 0 \\
4 & 0 & 0 & 2^{2m} + 4 & \cdots & 0 & 2(m-1)2^m + 3 & 1 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 2^{2m} - 3 & 4 & 2^{m} - 3 & 0 & \cdots & 2m2^m - 3 & 0 & 0 \\
0 & 2^{2m} - 2 & 4 & 2^{m} - 2 & 0 & \cdots & 2m2^m - 2 & 0 & 1 \\
2^{2m} - 1 & 0 & 4 & 2^{m} - 1 & 0 & \cdots & 2m2^m - 1 & 0 & 0 \\
2^{2m} & 0 & 4 & 2^{m} & 0 & \cdots & 2m2^m & 0 & 1 \\
\end{array}
\]

Table 2: Example Relation for FIs

\(^2\)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 \( A_{2i} \) for \( 1 \leq i \leq m \) with \( t[X] /= t'[X] \neq 0 \). 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 \( X \subseteq X' \) and \( X' \not\rightarrow A_{2m+1} \). This can be seen by the fact that each \( A_j \not\in X \) has a unique value. Hence, \( X' \rightarrow A_{2m+1} \) and \( \Sigma' \) is a minimum cover.

If the relation of Mannila and Räähä 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 \( \Sigma \) is transformed into a set \( \Sigma_m \) of FDs. This new set is called most general and can be computed with the following algorithm:

\[
\text{for each } X \rightarrow Y \in \Sigma \text{ do }
\begin{align*}
\quad & \text{if } Y = \text{closure}(X, \Sigma) \setminus X \text{ then } \\
\quad & \quad \Sigma_m := \Sigma_m \cup \{X \rightarrow Y\}
\end{align*}
\]

Obviously, the complexity depends on the cardinality of \( \Sigma \) 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 \((d_1, \ldots, d_{n+m+1})\) 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:
\[ \Sigma_n := \{ \} \]

\begin{verbatim}
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 \( A_i = a_i, \ldots, A_n = a_n \)
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 \( \nu \) with \( C_{i_1}, \ldots, C_{i_\nu} \) and \( \bigwedge_{i=1, \ldots, \nu} d_{i_\nu} \neq c_{i_\nu} \) then
\( \Sigma_n := \Sigma_n \cup A_1, \ldots, A_n, C_{i_1}, \ldots, C_{i_\nu} \rightarrow B_1, \ldots, B_m \)
end
end
\end{verbatim}

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 \( t_1[A_1, \ldots, A_n] \neq t_2[A_1, \ldots, A_n] \) and \( t_1[B_1, \ldots, B_m] \neq t_2[B_1, \ldots, B_m] \). We know by construction that \( t_2[C_{i_1}, \ldots, C_{i_\nu}] \neq t_2[C_{i_1}, \ldots, C_{i_\nu}] \). Hence, \( t_1[X, C_{i_1}, \ldots, C_{i_\nu}] \neq t_2[X, C_{i_1}, \ldots, C_{i_\nu}] \) and \( A_1, \ldots, A_n, C_{i_1}, \ldots, C_{i_\nu} \rightarrow B_1, \ldots, B_m \) is valid. \( \square \)

Completeness can now be seen by the following lemma:

**Lemma 10** Let \( \Sigma \) be the former set of FDs and \( \Sigma_n \) 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 \( \Sigma \) and the result is \( \Sigma' \), then \( r' \models \Sigma' \) by the lemmas. \( \square \)

### 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 \( \bigwedge_{i=1}^{n} b_i = b'_i \) 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:

\[
U1: (\text{Reflexivity}) \quad R[A] \supseteq R[A] \\
U2: (\text{Transitivity}) \quad \frac{R[A] \supseteq S[B], S[B] \supseteq T[C]}{R[A] \supseteq T[C]}
\]

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 \( a \not\in R[A] \).
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 $R[A] \supseteq S[B] \in \text{Cn}(I \cup I')$ iff $R[A] \supseteq S[B] \in \text{Cn}(I)$

**Proof:** (if) is trivial by the monotonicity of $\models$.

(only-if) We know that $R[A] \supseteq S[B] \in \text{Cn}(I \cup I)$ and $I \cup I'$ is consistent. If $R[A] \not\supseteq S[B]$ 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 $R[A] \supseteq S[B] \not\in \text{Cn}(I)$, 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, $R[A] \supseteq S[B] \in \text{Cn}(I)$.

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. $R[A] \supseteq S[B] \in \text{Cn}(I')$ 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 $R[A] \not\supseteq S[B] \not\in I'$. 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 $T[C] = 1$.

<table>
<thead>
<tr>
<th>T</th>
<th>...</th>
<th>C</th>
<th>...</th>
<th>U</th>
<th>...</th>
<th>D</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

$U 1 1 : \quad \frac{R[A] \supseteq S[B], R[A] \supseteq T[C]}{S[B], T[C]}$

$U 1 2 : \quad \frac{R[A] \supseteq T[C], S[B] \supseteq T[C]}{S[B], R[A]}$

**Theorem 7** The rules are sound.

**Proof:** Can easily be seen by contraposition of U2
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 \not
\supseteq B \not

\in 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 \( R[A] \not
\supseteq S[B] \) then the transitivity rule is the only one, which we can use to infer from \( I \cup \{ R[A] \supseteq S[B] \} \) the UIND \( T[C] \not
\supseteq U[D] \). This can be done with a chain in \( I \) :

\[ \{ T[C] \supseteq V[X_1], V[X_1] \supseteq V'[X_2], \ldots, V''[X_n] \supseteq R[A], S[B] \supseteq V'''[X_{n+1}], V''''[X_{n+2}] \supseteq T[D] \} \]

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.
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].

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

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

It takes \( O(n^2) \) time for closure and \text{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].

\[
CD: \begin{align*}
A_0 \rightarrow A_1, & A_1 \supseteq A_2, \ldots, A_{k-1} \rightarrow A_k, A_k \supseteq A_0 \\
A_1 \rightarrow A_0, & A_2 \supseteq A_1, \ldots, A_k \rightarrow A_{k-1}, A_0 \supseteq A_k
\end{align*}
\]

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 Fls.

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:

\[
CDI: \begin{align*}
A_0 \rightarrow A_1, & A_1 \supseteq A_2, \ldots, A_{k-1} \rightarrow A_k, A_k \supseteq A_0, A_{i+10'}A_i \\
A_1 \rightarrow A_0, & A_2 \supseteq A_1, \ldots, A_k \rightarrow A_{k-1}, A_0 \supseteq A_k
\end{align*}
\]

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 \( \text{card}(A_1) \geq \text{card}(A_2) \). This means that \( \text{card}(A_{i+1}) \geq \text{card}(A_i) \). If \( i = l \) and \( l \) is even it is the case that \( \text{card}(A_1) \nmid \text{card}(A_{i+1}) \). This means \( \text{card}(A_{i+1}) > \text{card}(A_i) \) and therefore \( A_l \nmid A_{i+1} \). If \( i \neq l \) then there is the case that \( \text{card}(A_i) \geq \text{card}(A_{i+1}) \) and \( \text{card}(A_{i+1}) \nmid \text{card}(A_i) \). It follows that \( \text{card}(A_i) > \text{card}(A_{i+1}) \) and that \( \text{card}(A_{i+1}) > \text{card}(A_i) \). This means that for the corresponding o it is the case that \( A_1 \nmid A_{i+1} \) or \( A_l \nmid A_{i+1} \).

The sets of FDs and UNIDs are partitioned to a new set called DEP and the FI and UNIs 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. \( A_i \cdot \rho A_i \in Cn(INDEP \cup DEP) \) if and only if \( A_i \cdot \rho A_i \in Cn(DEP) \) with \( o = \rightarrow \) or \( o = \geq \).

**Proof:** (if) trivial.

(only-if) Case 1) \( o = \rightarrow \): This means that \( A \rightarrow B \in Cn(DEP \cup INDEP) \). Assume that \( A \rightarrow B \not\in Cn(DEP) \), 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. \( A \cdot \rho B \in Cn(INDEP) \) if and only if \( A \cdot \rho B \in INDEP \) with \( o = \rightarrow \) or \( o = \geq \).

**Proof:** (if) trivial.

(only-if, sketch) Assume that \( A \cdot \rho B \in Cn(INDEP) \) and \( A \cdot \rho B \not\in INDEP \). 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 \( DEP \cup INDEP \models A_i \nmid A_{i+1} \) iff \( DEP \cup INDEP \cup \{ A_i \nmid A_{i+1} \} \) is inconsistent whereas \( o \) stands for \( \rightarrow \) or \( \geq \) and \( A_i \) and \( A_{i+1} \) are some attributes.

**Proof:** (only-if) is trivial.

(if) By lemma the assumption can be transformed to \( A_{i+1} \nmid' A_i \in Cn(DEP \cup \{ A_i \nmid A_{i+1} \}) \) for some \( A_{i+1} \nmid' A_i \in INDEP \), whereas \( A_{i+1} \) and \( A_i \) are some attributes. A stronger result is shown that this implies \( DEP \cup \{ A_{i+1} \nmid' A_i \} \models A_i \nmid A_{i+1} \). Assume \( DEP \cup INDEP \) is consistent. This implies \( A_i \nmid A_{i+1} \not\in Cn(DEP) \). There are four cases: (We denote with \( \Sigma \) the set of FDs.)

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

**ad 1**: case 1: If \( A_{i+1} \rightarrow A_i \in Cn(\Sigma \cup \{A_i \rightarrow A_{i+1}\}) \), then it is known by completeness of FDs and FIs that \( DEP \cup \{A_{i+1} \rightarrow A_i\} \vdash A_i \rightarrow A_{i+1} \).

case 2: If \( A_{i+1} \rightarrow A_i \not\in Cn(\Sigma \cup \{A_i \rightarrow A_{i+1}\}) \), then it is known by completeness of DEP, that \( DEP \cup \{A_i \rightarrow A_{i+1}\} \vdash A_{i+1} \rightarrow A_i \) and this means the inference rule \( CD \) can be applied and therefore \( A_0 \rightarrow A_1, \ldots, A_{i-1} \rightarrow A_i \rightarrow A_{i+1}, \ldots, A_k \rightarrow A_0 \in Cn(DEP) \). By II it can be inferred \( DEP \cup \{A_{i+1} \rightarrow A_i\} \vdash A_i \neq A_{i+1} \).

**ad 2**: Assume that \( A_0 \rightarrow A_1, \ldots, A_{i-1} \supseteq A_i, A_{i+1} \supseteq A_{i+2}, \ldots, A_k \supseteq A_0 \in Cn(DEP) \). Therefore it can be inferred from \( DEP \vdash A_0 \rightarrow A_1, \ldots, A_{i-1} \rightarrow A_i, A_{i+1} \rightarrow A_{i+2}, \ldots, A_k \rightarrow A_0 \) and from \( DEP \cup \{A_{i+1} \backslash A_i\} \vdash A_i \neq A_{i+1} \).

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 a 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: \( \frac{A \rightarrow B}{|A| \geq |B|} \)
- N4: \( \frac{A \rightarrow B, |B| > |A|}{B \rightarrow A} \)
- N5: \( \frac{A \rightarrow B, |B| \geq |A|}{B \rightarrow A} \)
- N6: \( \frac{A \rightarrow B, |B| > |A|}{B \rightarrow A} \)
- N7: \( \frac{R \rightarrow A : S \rightarrow B}{S[B] \geq |P[A]|} \)

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.

- Rule 11: \[ \frac{|A| \geq |B|}{A \not\in B} \]
- Rule 12: \[ \frac{|A| \geq |B|}{A \notin B} \]
- Rule 13: \[ \frac{|A| \geq |B|}{B \notin A} \]
- Rule 14: \[ \frac{|A| \geq |B|}{B \geq A} \]

- Rule 15: \[ \frac{|A| \geq |B|}{A \geq |B|} \]
- Rule 16: \[ \frac{A \subseteq B, B \not\subseteq A}{|A| \geq |B|} \]
- Rule 17: \[ \frac{A \subseteq B, B \not\subseteq A}{A \geq |B|} \]
- Rule 18: \[ \frac{|A| \geq |B|, |B| \geq |C|}{|A| \geq |C|} \]
- Rule 19: \[ \frac{|A| \geq |B|, |B| \geq |C|}{|A| \geq |C|} \]

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

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 \( DEP \cup INDEP \models AoB \) 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 table 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, 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$\not\rightarrow$ A, I, I') then succeed
else if Infer-FI(A$\not\rightarrow$ B, I, I') then succeed
end
else
if B > A then succeed
else if B ≥ A and Infer-UINI(B$\not\rightarrow$ A, I, I') then succeed
else if Infer-UINI(A$\not\rightarrow$ B, I, I') then succeed
end

Table 5: Membership Algorithm for FIs and UINIs

Definition 16 (Multivalued Dependency (MVD)) The MVD $X \rightarrow Y$ 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) $t_3[X] = t_1[X]$, $t_3[Y] = t_1[Y]$ and $t_3[Z] = t_2[Z]$
(ii) $t_4[X] = t_2[X]$, $t_4[Y] = t_2[Y]$ and $t_4[Z] = t_1[Z]$

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 $X \rightarrow U - X$

$MDa$ : (Augmentation) $X \rightarrow Y$ $Z \rightarrow Y$

$MDd$ : (Difference) if $Y \cap Z = \emptyset$ then $X \rightarrow Y, Z \rightarrow Y$ $Z \rightarrow Y$

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 $X \not\rightarrow Y$ 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) $t_3[X] = t_1[X]$, $t_3[Y] = t_1[Y]$ and $t_3[Z] = t_2[Z]$

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:

$MI_r : \frac{XZ \not\in Y}{X \not\in Y}$

$MI_1 : \text{if } Y \cap Z = \emptyset \text{ then } \frac{X \rightarrow Y}{Z \not\in Y}$

$MI_2 : \text{if } Y \cap Z = \emptyset \text{ then } \frac{X \rightarrow Y}{Z \not\in Y}$

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:

$MFD_t : \frac{X \rightarrow Y}{X \not\rightarrow Y}$

$MFD_i : \text{if } Y \cap Z = \emptyset \text{ then } \frac{X \rightarrow Y \cap Z}{X \not\rightarrow Y}$

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

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

$MFI_1 : \frac{X \not\in Y}{X \not\in Y}$

$MFI_2 : \text{if } Y \cap Z = \emptyset \text{ then } \frac{X \rightarrow Y \not\in Z}{Z \not\in Y}$

$MFI_3 : \text{if } Y \cap Z = \emptyset \text{ then } \frac{Z \not\in Y}{X \not\in Y}$

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 \( \Theta \)-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 presume 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


