Approximate inference of functional dependencies from relations

Jyrki Kivinen, Heikki Mannila

Department of Computer Science, University of Helsinki, P.O. Box 26, Teollisuuskatu 23,
FIN-00014 Helsinki, Finland

Abstract

The functional dependency inference problem is the following. Given a relation \( r \), find a set of functional dependencies that is equivalent with the set of all functional dependencies holding in \( r \). All known algorithms for this task have running times that can be in the worst case exponential in the size of the smallest cover of the dependency set. We consider approximate dependency inference. We define various measures for the error of a dependency in a relation. These error measures have the value 0 if the dependency holds and a value close to 1 if the dependency clearly does not hold. Depending on the measure used, all dependencies with error at least \( \varepsilon \) in \( r \) can be detected with high probability by considering only \( O(1/\varepsilon) \) or \( O(|r|^{1/2}/\varepsilon) \) random tuples of \( r \). We also show how a machine learning algorithm due to Angluin et al. can be applied to give in output-polynomial time an approximately correct cover for the set of dependencies holding in \( r \).

1. Introduction

In database design, integrity constraints are conditions that define what database states are allowed. There exist several classes of dependencies (see, e.g., [16, 30, 20]). In practice, the most important class consists of functional dependencies. Given a set of attributes \( R \), a functional dependency over \( R \) is an expression \( X \rightarrow Y \), where \( X, Y \subseteq R \). If \( r \) is a relation over \( R \), i.e., a finite set of mappings (called rows or tuples) from \( R \) to some domain, then the dependency \( X \rightarrow Y \) holds in \( r \), or \( r \) satisfies \( X \rightarrow Y \), if all pairs of tuples that agree on \( X \), agree also on \( Y \). Denoting by \( t[Z] \) the restriction of a tuple \( t \in r \) to the set of attributes \( Z \subseteq R \), the notion of satisfaction can be defined as: for all \( t, t' \in r \), if \( t[X] = t'[X] \), then \( t[Y] = t'[Y] \). Throughout this paper, we consider only functional dependencies and call them just dependencies. If \( X \rightarrow Y \) holds in \( r \), we write \( r \models X \rightarrow Y \).
Given a relation $r$ over a schema $R$, let

$$\text{dep}(r) = \{X \rightarrow Y \mid X, Y \subseteq R, \ r \models X \rightarrow Y\}$$

be the set of functional dependencies that hold in $r$. A cover for a set $F$ of dependencies is a set $G$ of dependencies such that $F$ and $G$ are equivalent, i.e., any relation that satisfies all the dependencies of $F$ also satisfies all the dependencies of $G$ and vice versa. The dependency inference problem is the problem of finding a small cover for $\text{dep}(r)$. This problem has applications in database design [29, 17, 6], analysis of existing databases [7], query optimization [8, 28] and artificial intelligence [25, 26, 1]. See [24] for some other applications of rule inference from database instances.

The problem of inferring functional dependencies from a relation $r$ has been shown to require time $\Omega(|r| \log |r|)$ even for a schema containing two attributes. Furthermore, for each $n$ there exists a relation $r$ over a schema of $n$ attributes such that $|r| = O(n)$, but each cover of $\text{dep}(r)$ has $\Omega(2^{n^2})$ dependencies [18, 19]. Hence, there is no hope of achieving a polynomial-time solution for the problem.

What remains open, however, is the existence of an algorithm operating in polynomial time with respect to the size of the relation and the size of the smallest cover for the dependency set of the relation. Previous algorithms for the dependency inference problem all require in the worst case exponential time with respect to the size of the output. In fact, it has been shown [12] that the existence of an output-polynomial algorithm for dependency inference would imply the existence of a similar algorithm for several other open problems.

Since the number of tuples $|r|$ in the relation is typically much larger than the number of attributes, a useful property of practical dependency inference algorithms is that the running time has the form $O(|r| \log |r|)$ for fixed $n$. This is a sensible goal also because checking whether a dependency $X \rightarrow Y$ holds in $r$ can be done by sorting $r$ with respect to the $X$ values.

Hence, the goal of the development of dependency inference algorithms is to find a method using time $O(q(|H|)\text{sort}(r))$, where $q$ is a polynomial, $H$ is the smallest cover for the dependencies holding in $r$ and $\text{sort}(r) = O(n|r| \log |r|)$ is the time needed for sorting the relation $r$.

In this paper we consider approximate dependency inference, that is, dependency inference where the result need not be completely accurate. The paper contains two types of results.

We define in Section 2 some measures $g$ for the error $g(f, r)$ of a dependency $f$ in a relation $r$. If $f$ holds in $r$, then $g(f, r) = 0$. If $f$ clearly does not hold, then $g(f, r) \approx 1$. In Section 3 we show that depending on which measure $g$ we use, already a set of either $O(1/\varepsilon)$ or $O(|r|^{1/2}/\varepsilon)$ tuples, chosen from $r$ randomly, violates with high probability every dependency $f$ with $g(f, r) > \varepsilon$.

In Section 4 we modify an algorithm for learning conjunctions of Horn clauses, due to Angluin et al. [3]. We get a randomized algorithm for finding, with high probability, a set $F$ of dependencies such that $d(F, \text{dep}(r)) < \varepsilon$ where $d$ is a certain distance measure for dependency sets. The algorithm works in polynomial time with respect to...
and the size of the smallest cover for $\text{dep}(r)$, and the running time is $O(|r| \log |r|)$ with respect to $|r|$. Thus, the algorithm achieves the goals above at the expense of allowing a small error. Similar results, on slightly different frameworks, have been obtained independently by Dechter and Pearl [9] and Kautz et al. [13].

We assume the reader is familiar with the basic notions of functional dependencies [30, 16]. Here we just mention some notational conventions. If $A \in R$ is a single attribute, we write, for example, $Z \rightarrow A$ instead of $Z \rightarrow \{A\}$. Similarly, we use $Z+: to denote $Z \cup \{A\}$, and $XY$ to denote $X \cup Y$.

2. Error measures for dependencies

We would like to have a formal definition for having a dependency almost hold in a relation. Besides a wish to explicate this seemingly natural intuitive notion, we have some specific applications in mind. As we shall see in Section 3, we can determine dependencies quite reliably from a random sample of the relation. However, with a reasonable sample size it is not possible to distinguish exactly true dependencies from those that almost hold.

We suggest three measures, $G_1$, $G_2$ and $G_3$, for measuring the error of a dependency $X \rightarrow Y$ in a relation $r$. Their respective scaled versions $g_1$, $g_2$ and $g_3$ all range over $[0, 1]$, having the value 0 when $X \rightarrow Y$ holds in $r$ and the value 1 (or almost 1) if for all tuples $u, v \in r$ such that $u \neq v$, we have $u[X] = v[X]$ but $u[Y] \neq v[Y]$. Thus, we might say that $X \rightarrow Y$ almost holds if some of the measures $g_i$ has value close to 0. However, we shall see that for some relations the measures give very different values. It is not clear which of them, if any, is the most natural measure for the degree of truth of a dependency.

Consider now a relation $r$ and a functional dependency $X \rightarrow Y$ over $R$. We say that a pair $(u, v)$ of tuples of $r$ violates the dependency, or is a violating pair for it, if $u[X] = v[X]$ but $u[Y] \neq v[Y]$. Hence, a dependency holds in a relation if the relation does not contain violating pairs for it. A single tuple $u$ is called violating if it is a component in some violating pair. We define the three measures $G_1$, $G_2$ and $G_3$ to be the number of violating pairs, the number of violating tuples and the number of tuples one has to delete to obtain a relation that satisfies the dependency, respectively. The corresponding scaled measures are denoted by $g_1$, $g_2$ and $g_3$. The scaling factor is the number of tuple pairs $|r|^2$ for the $g_1$ measure, and the number of tuples $|r|$ for the $g_2$ and $g_3$ measures. More formally, we define

$$G_1(X \rightarrow Y, r) = |\{(u, v) \mid u, v \in r, u[X] = v[X], u[Y] \neq v[Y]\}|,$$

$$g_1(X \rightarrow Y, r) = G_1(X \rightarrow Y, r)/|r|^2,$$

$$G_2(X \rightarrow Y, r) = |\{u \mid u \in r, \exists v \in R : u[X] = v[X], u[Y] \neq v[Y]\}|,$$

$$g_2(X \rightarrow Y, r) = G_2(X \rightarrow Y, r)/|r|,$$

$$G_3(X \rightarrow Y, r) = |r| - \max\{|s| \mid s \subseteq r, s \models X \rightarrow Y\} \quad \text{and}$$

$$g_3(X \rightarrow Y, r) = G_3(X \rightarrow Y, r)/|r|. $$
One can compute the values of \( G_1, G_2 \) and \( G_3 \) in time \( O(sort(r)) \).

The relationships between these different measures are discussed in detail in Appendix A. Here we just note that the scaling factor is \( |r|^2 \) for the measure \( g_1 \) but \( |r| \) for the measures \( g_2 \) and \( g_3 \). Thus, as the next example shows, it is possible that the measure \( g_1 \) gets a much smaller value than the measures \( g_2 \) and \( g_3 \).

**Example 2.1.** Given an arbitrary fraction \( \varepsilon = q/p < \frac{1}{2} \), we construct a relation \( r \) over \( BA \) with \( |r| = p \) such that \( g_3(B \rightarrow A, r) = \varepsilon \), \( g_2(B \rightarrow A, r) = 2\varepsilon \) and \( g_1(B \rightarrow A, r) = 2\varepsilon/p \). The relation \( r \) contains the tuple \((j,0)\) for \( j = 1, \ldots, p - 2q \) and the tuples \((j,1)\) for \( j = p - 2q + 1, \ldots, p - q \). Then \( G_1(B \rightarrow A, r) = G_2(B \rightarrow A, r) = 2q \) and \( G_3(B \rightarrow A, r) = q \).

In Appendix A we also consider inferring new dependencies using Armstrong's inference rules. We allow the dependencies taken as premises to have a nonzero error according to one of the measures \( g_i \). It turns out that the error of a dependency obtained by applying an inference rule can be as large as the sum of the errors of the dependencies used as premises, but never larger.

An alternative measure for the truth of a data dependency has been proposed by Piatetsky-Shapiro [23]. It is not directly equivalent to any of the above measures, but some bounds can be obtained. We omit the details.

Next we define a measure for the distance between two sets of functional dependencies over \( R \), without reference to any particular relation over \( R \). Note that it is not easy to relate such measures to the measures \( g_i \), since the values of measures \( g_i \) can vary widely in relations in which the same dependencies hold exactly.

Let \( F \) be a set of functional dependencies over \( R \). The closure of a subset \( X \subseteq R \) under \( F \) is the unique maximal subset \( Y \subseteq R \) such that \( X \not\subseteq Y \) holds in every relation that satisfies all the dependencies of \( F \). Equivalently, the closure of \( X \) is the set

\[
\{ B \in R \mid \text{for all } r : \text{if } r \models F, \text{ then } r \models X \rightarrow B \}.
\]

We say that the subset \( X \) is \( F \)-closed if the closure of \( X \) under \( F \) is \( X \). The collection of all \( F \)-closed subsets of \( R \) is denoted by \( \text{CL}(F) \). Clearly, \( \text{CL}(F) = \text{CL}(G) \) if and only if \( F \) and \( G \) are equivalent. Thus, one can identify the set of dependencies \( F \) with \( \text{CL}(F) \), which is a subset of the set \( \mathcal{P}(R) \) (the collection of subsets of \( R \)). Let \( A \triangle B \) denote the symmetrical difference \((A \cup B) - (A \cap B)\). For any probability measure \( P \) on \( \mathcal{P}(R) \) we now define the function \( d_P \) by

\[
d_P(F,G) = P(\text{CL}(F) \triangle \text{CL}(G)).
\]

It is clear that \( d_P \) is a metric on equivalence classes of dependency sets (assuming \( P(A) > 0 \) for all nonempty \( A \subseteq \mathcal{P}(R) \)); \( d_P \) is also called the Mazurkiewicz metric [22]

In dependency inference we are interested in finding a small cover for \( \text{dep}(r) \). In approximate dependency inference, we merely wish to find a set \( F \) that is close to
dep(r) as measured by $d_p$. In Section 4 we show how we can with high probability produce a dependency set $F$ such that $d_p(F, \text{dep}(r)) \leq \varepsilon$.

We can define an alternative measure $d'_p(F, G)$ of distance between dependency sets as the probability that for a random subset $X$ the closure $X_F^*$ of $X$ under $F$ is different from the closure of $X$ under $G$. That is,

$$d'_p(F, G) = P\{X \subseteq R \mid X_F^* \neq X_G^*\}.$$  

This is again a metric on equivalence classes of dependency sets. The measure $d'_p$ mirrors the view of dependency sets as closure operations, whereas the measure $d_p$ views dependency sets as defining the predicate "is a closed set".

3. Finding approximate dependencies by random sampling

Since dependency inference seems to be computationally difficult and the relations are usually large, it would be desirable to be able to determine the dependencies in a relation $r$ by considering just a small subset $s$ of $r$. In this section we study how large a subset we must consider in order to get reasonably good results. We assume that the subrelation $s$ is obtained by random sampling from $r$. Therefore, there is always at least a small possibility that a dependency holds in $s$ but not in $r$. If the dependency is clearly erroneous, we can hope to make the probability of this event low.

Let $g$ be one of the measures $g_i$ considered in Section 2. Given a parameter $0 < \varepsilon < 1$, we say that a dependency $X \rightarrow Y$ is $\varepsilon$-good with respect to $g$ in $r$ if $g(X \rightarrow Y, r) \leq \varepsilon$. Otherwise the dependency is $\varepsilon$-bad.

A sequence $s$ consisting of $m$ tuples from the relation $r$ is a sample of $r$. The parameter $m$ is the size of the sample. We assume that samples are obtained by choosing tuples at random, with replacement, from the uniform probability measure on $r$. We say that $X \rightarrow Y$ holds in $s$, and write $s \models X \rightarrow Y$, if $X \rightarrow Y$ holds in the relation $s'$ that consists of all the tuples in $s$. (Note that $s$ itself is not a relation but an ordered sequence that may contain duplicates.)

The results of this section give bounds for the sample size $m$ in terms of an accuracy parameter $\varepsilon$ and a confidence parameter $\delta$. We would like $m$ to be bounded by a polynomial of $1/\varepsilon$ and $1/\delta$. We may also need to allow the size $|r|$ of the relation to affect the sample size. If a dependency $X \rightarrow Y$ is $\varepsilon$-bad, then for a sample $s$ of size $m$ the probability of having $s \models X \rightarrow Y$ should be at most $\delta$. Thus, we have a high confidence in detecting clearly erroneous dependencies. If the dependency is $\varepsilon$-good, we allow the dependency to hold in $s$, even if it does not hold in $r$. Obviously, if a dependency does not hold in $s$, it cannot hold in $r$.

The idea of using random sampling and controlling the quality of the approximation by accuracy and confidence parameters is similar to the approach proposed by Valiant in machine learning [31, 2].

We start by looking at one dependency at a time. Consider the following algorithm.
Algorithm 3.1. Using a sample to determine whether a dependency $X \rightarrow Y$ is $\epsilon$-good with respect to $g_2$ in $r$.

Input. A relation $r$, an error parameter $\epsilon$ and a confidence parameter $\delta$.

Output. “yes” if $r \models X \rightarrow Y$; “no” with probability at least $1 - \delta$ if $g_2(X \rightarrow Y, r) > \epsilon$.

Method.

$m := \left\lceil \left(\frac{1}{\epsilon^2} \ln\left(\frac{1}{\delta}\right)\right) \right\rceil$;

repeat $m$ times
  $u :=$ random tuple from $r$;
  $S := \{ v[Y] \mid v \in r, v[X] = u[X] \}$;
  if $|S| > 1$ then return “no”;
end repeat;
return “yes”.

Algorithm 3.1 is similar in spirit to the method of Lipton et al. [15] for estimating the size of a query by sampling. In our special case, a much simpler analysis is sufficient. The inequality

$$1 - x \leq e^{-x}$$

that holds for all $x$ is repeatedly used in the proofs that follow.

Proposition 3.2. If the dependency $X \rightarrow Y$ is $\epsilon$-bad with respect to $g_2$ in $r$, then Algorithm 3.1 returns “no” with probability at least $1 - \delta$.

Proof. The algorithm returns “yes” if and only if none of the $m$ random tuples $u$ appear in a violating pair. Under our assumptions, the probability of this event is

$$(1 - g_2(X \rightarrow Y, r))^m < (1 - \epsilon)^m \leq e^{-\epsilon m} \leq \delta.$$  \hfill \Box$

The amount of random sampling done by Algorithm 3.1 is very reasonable. For instance, if we wish to have a 95% confidence in detecting the cases in which the dependency is 10% bad with respect to $g_2$, we have $m = 30$, no matter how large $r$ is. The algorithm also avoids the computational problem of checking the validity of the dependency in a subrelation. The drawback of the method is that the computations it performs depend on the particular $X$ we consider. Therefore, we must apply the algorithm separately for each dependency we are interested in.

Next we consider whether a single sample could be used to infer all the $\epsilon$-bad dependencies in the relation with reasonable confidence. That is, we would like to take a single sample $s$ in such a way that with probability at least $1 - \delta$, no dependency that is $\epsilon$-bad in $r$ holds in $s$. After drawing the sample $s$, we infer the dependencies holding in it using any exact dependency inference algorithm [21]. Assuming the sample is small enough, the complexity of the exact algorithms does not matter. At the end of this section we discuss how one can base an exact dependency inference algorithm for the original relation on the use of a sample.

Using sampling in detecting all $\epsilon$-bad dependencies with respect to $g_2$ seems hopeless. For instance, consider the relation $r$ over $ABC$ that contains the tuple $(|r|, 1, 0)$...
and for $j = 1, \ldots, |r| - 1$ the tuple $(j, 1, 1)$. We then have $g_2(B \rightarrow C, r) = 1$. However, for a sample $s$ of $r$ we have $s \models B \rightarrow C$ unless the tuple $(|r|, 1, 0)$ is in $s$. The probability of having a given tuple in $s$ is vanishing unless the sample size is $\Omega(|r|)$.

We next derive bounds for the sample size required in order to detect all the dependencies that are $\varepsilon$-bad with respect to $g_1$ or $g_3$. We start with $g_1$ and only one dependency.

**Proposition 3.3.** If the dependency $X \rightarrow Y$ is $\varepsilon$-bad with respect to $g_1$ in $r$, then the probability of drawing from $Y$ a sample $s$ of size $m$ such that $X \rightarrow Y$ holds in $s$ is at most $\delta$ for

$$m \geq 2 \left\lceil \frac{1}{\varepsilon} \ln \frac{1}{\delta} \right\rceil .$$

**Proof.** If the tuples number $2i - 1$ and $2i$ in the sequence $s$ violate the dependency, then certainly the dependency does not hold in $s$. The probability of the event that a given pair does not violate the dependency is $1 - g_1(X \rightarrow Y, r) < 1 - \varepsilon$. We can divide $s$ into $m/2$ independent pairs, and the probability that none of these violates the dependency is

$$(1 - g_1(X \rightarrow Y, r))^{m/2} \leq (1 - \varepsilon)^{m/2} \leq e^{-\varepsilon m/2} \leq \delta .$$

It is important that the bound of Proposition 3.3 depends only logarithmically on the confidence parameter $\delta$. This allows us to consider all the dependencies at once and obtain the following result.

**Corollary 3.4.** Let $s$ be a sample of a relation $r$ over $n$ attributes, with the sample size $m$ satisfying

$$m \geq 2 \left\lceil \frac{1}{\varepsilon} \left( 2n \ln 2 + \ln \frac{1}{\delta} \right) \right\rceil .$$

With probability at least $1 - \delta$, no dependency that is $\varepsilon$-bad with respect to $g_1$ in $r$ holds in $s$.

**Proof.** For any given dependency $X \rightarrow Y$ that is $\varepsilon$-bad, the probability of the event $s \models X \rightarrow Y$ is by Proposition 3.3 at most $\delta/(2^n)$. Since there are at most $2^n$ such dependencies, the result follows.

Again, the sample size does not grow as a function of $|r|$. We can use one small sample to find all $\varepsilon$-bad dependencies with respect to $g_1$. One problem with this bound is that, as shown in Example 2.1 in Section 2, a dependency can be $\varepsilon$-good with respect to $g_1$ and still $\varepsilon'$-bad with respect to $g_2$ or $g_3$ for $\varepsilon/\varepsilon' \leq 1$. If we want to detect $\varepsilon$-bad dependencies with respect to $g_3$, we have a lower bound $\Omega(|r|^{1/2})$ for the required sample size.
Proposition 3.5. For every rational value $0 < \varepsilon < \frac{1}{2}$ there are arbitrarily large relations $r$ such that the dependency $B \rightarrow A$ is $\varepsilon$-bad with respect to $g_3$ in $r$ but holds with probability at least $\frac{1}{2}$ in a sample of size less than $\left(\frac{|r|}{(2\varepsilon)^{1/2}}\right)^{1/2}$.

Proof. Take an arbitrary fraction $\varepsilon = \frac{q}{p} < \frac{1}{2}$. We consider the relation $r$ that was used in Example 2.1 to show that we can keep $g_3(B \rightarrow A, r) = \varepsilon$ while making $g_1(B \rightarrow A, r)$ arbitrarily close to 0. Thus, let the relation $r$ over $BA$ contain the tuple $(j,0)$ for $j = 1, \ldots, p-2q$ and the tuples $(j,0)$ and $(j,1)$ for $j = p-2q+1, \ldots, p-q$. Then $|r| = p$ and $g_3(B \rightarrow A, r) = \varepsilon$.

The probability that a random sequence of tuples contains two given tuples in two given positions is $1/p^2$. In a sample of size $m$, there are $m(m-1)/2$ possible locations for a given pair of tuples, so the probability that given two tuples appear somewhere in the sample is less than $m^2/p^2$. A sample $s$ of $r$ with $s \nleq B \rightarrow A$ must contain, for some $j$, both the tuple $(j,0)$ and the tuple $(j,1)$. Since there are $q$ possible values for $j$, the probability of this is at most $qm^2/p^2$ for sample size $m$. This value is less than $\frac{1}{2}$ for

$$m < \left(\frac{p^2}{2q}\right)^{1/2} = \left(\frac{|r|}{2\varepsilon}\right)^{1/2}.$$ 

The following result shows that the $\Omega(|r|^{1/2})$ lower bound of Proposition 3.5 can be achieved.

Theorem 3.6. If the dependency $X \rightarrow Y$ is $\varepsilon$-bad with respect to $g_3$ in $r$, then the probability of drawing from $r$ a sample $s$ with size $m$ such that $X \rightarrow Y$ holds in $s$ is at most $\delta$ for

$$m \geq \max \left\{ \frac{8}{\varepsilon} \ln \frac{2}{\delta}, \frac{2}{\varepsilon} \left[ \frac{\log(2/\delta)}{\log(4/3)} \right] \left(2|s| \ln 2 \right)^{1/2} + 1 \right\}$$

$$= O \left(\frac{|r|^{1/2} \ln \frac{1}{\delta}}{\varepsilon \log \frac{1}{\delta}}\right).$$

Before going into the proof of Theorem 3.6, we note that idea of the proof of Corollary 3.4 is again applicable. Thus, we see that one sample can be used for all the dependencies.

Corollary 3.7. Let $s$ be a sample of size $m$ from a relation $r$ over $n$ attributes. Then with probability at least $1 - \delta$, no dependency that is $\varepsilon$-bad with respect to $g_3$ in $r$ holds in $s$, provided that the sample size $m$ satisfies

$$m \geq \max \left\{ \frac{8}{\varepsilon} \left(2n \ln 2 + \ln \frac{2}{\delta}\right), \frac{2}{\varepsilon} \left[ \frac{2n \log 2 + \log(2/\delta)}{\log(4/3)} \right] \left((2|s| \ln 2 \right)^{1/2} + 1 \right\}$$

$$= O \left(\frac{|r|^{1/2} \ln \frac{1}{\delta}}{\varepsilon \log \frac{1}{\delta}}\right).$$

Proof of Theorem 3.6. We create a balls and urns model of $r$ by associating a ball to each tuple of $r$. We use the value combinations of the attributes in $X$ as names
for urns and the value combinations of the attributes in \( Y \) as \textit{colours}. We put the ball associated with the tuple \( u \) into the urn \( u[X] \) and give it the colour \( u[Y] \). Sampling now means choosing a sequence \( s \) of balls at random. The dependency \( X \rightarrow Y \) holds in \( s \) if and only if all the balls chosen from any given urn have the same colour.

Let \( \varepsilon \) and \( \delta \) be given, and let \( G = G_3(X \rightarrow Y, r) \). Assume that \( G \geq \varepsilon |r| \) and that \( m \) satisfies (1). We claim that with probability at least \( 1 - \delta \), a random sample of \( m \) balls contains two balls that are from the same urn but have different colour. This is equivalent to the statement of the theorem.

We now modify our model by employing additional urns and moving some of the balls from their original urns into the new ones. Each new urn will contain exactly two balls. These two balls have different colour and were also originally in same urn. From each original urn we remove pairs of differently coloured balls and put each pair into its own new urn, until all the balls remaining in the original urn have the same colour.

To prove our claim about the original balls and urns model, it is sufficient to show in the modified model that with probability at least \( 1 - \delta \), a sample of \( m \) balls contains both balls from one of the new urns. To achieve this, we first need to show that the sample is likely to contain a large number of balls from the new urns.

Let \( r[X] = \{ x_1, \ldots, x_p \} \) be the set of names of the original urns. For \( i = 1, \ldots, p \), let \( q_i \) be the largest number such that the urn \( x_i \) originally contained a set of \( q_i \) balls with the same colour. Then \( G = |r| - \sum_i q_i \). After the modification of the model, no more than \( q_i \) balls can remain in any original urn \( x_i \). Therefore, the total number of balls in the new urns must be at least \( |r| - \sum_i q_i = G \geq \varepsilon |r| \).

Let \( s = (b_1, \ldots, b_m) \) be a random sequence of \( m \) balls from the modified model. The probability of having \( b_i \) from one of the new urns is at least \( \varepsilon \) for all \( i \). Let the random variable \( L \) be the number of indices \( i \) such that the ball \( b_i \) is from a new urn. Then \( L \) can be considered the number of successes in \( m \) independent trials, each with probability at least \( \varepsilon \) of success. Hence, by the Chernoff bounds [4] for the binomial distribution, the probability of the event \( L \leq m\varepsilon /2 \) is at most \( e^{-mc/8} \). By substituting into this upper bound the lower bound \( m \geq (8/\varepsilon) \ln(2/\delta) \) from (1), we see that the probability of the event \( L \leq m\varepsilon /2 \) is at most \( \delta^2 /2 \).

We have shown that with a high probability, a reasonably large fraction of the sample consists of balls from the new urns. When this is the case, we can consider a subsample that contains only balls from the new urns and is still reasonably large. We then divide the subsample into a number of segments. We show that any given segment contains two different balls from the same new urn with probability at least \( 1/4 \). By having a large enough number of independent segments, we can have a very low probability for the event that none of them contain such two balls.

Thus, assume that \( L \geq m\varepsilon /2 \). By the second lower bound given in (1) for \( m \), this implies \( L \geq h/l \) where

\[
    h = \left\lceil \frac{\log(2/\delta)}{\log(\frac{4}{3})} \right\rceil
\]
and

\[ l = \left( \frac{\ln 2}{2} + 1 \right) . \]

We extract from \( s \) a subsample that consists of the first \( hl \) balls that are from a new urn. We then divide the subsample into \( h \) independent segments of length \( l \).

Consider now a fixed segment \( d = (d_1, \ldots, d_l) \). The balls \( d_i \) are drawn independently of each other from the population of balls that are in the new urns. Since every new urn contains exactly two balls, we can assume that the segment has been obtained by choosing first a sequence \( (u_1, \ldots, u_l) \) of \( l \) new urns and then for each \( i \) the ball \( d_i \) from the urn \( u_i \) at random.

Let \( k \) be the number of new urns. Assume first that \( l < k \). There are \( k^l \) possible choices for the sequence of the urns, and \( k(k - 1) \ldots (k - l + 1) \) choices such that \( u_i \neq u_j \) for all \( i \neq j \). Hence, the probability that no urn appears twice in the sequence is

\[ \frac{k(k - 1) \ldots (k - l + 1)}{k^l} \leq \left( 1 - \frac{l - 1}{2k} \right)^{l/2} \leq \exp \left( -\frac{l(l - 1)}{4k} \right) . \]

Since \( |r| \geq 2k \), we have \( l(l - 1) \geq 4k \ln 2 \), so this probability is at most \( \frac{1}{2} \). Hence, with probability at least \( \frac{1}{2} \), we have \( u_i = u_j \) for some \( i \neq j \). This is obviously true also if \( l \geq k \).

Under the condition that \( u_i = u_j \) where \( i \neq j \), the probability of having \( d_i \neq d_j \) is exactly \( \frac{1}{2} \). But under this condition, having \( d_i \neq d_j \) means that the segment contains the two different balls \( d_i \) and \( d_j \) from the urn \( u_i \). Since the condition is with probability at least \( \frac{1}{2} \) satisfied for some \( i \neq j \), the probability of the event that the segment contains two different balls from some urn is at least \( \frac{1}{4} \).

There are \( h \) independent segments, and for each segment the probability of the event that the segment fails to have two different balls from some urn is at most \( \frac{1}{2} \). Therefore, the probability of having this failure for all the \( h \) segments is at most \( \left( \frac{1}{2} \right)^h \leq \frac{\delta}{2} \).

We have seen that the probability of the having \( L \geq \varepsilon m \) but not having two different balls from any new urn is at most \( \frac{\delta}{2} \), and that the probability of not having \( L \geq \varepsilon m \) is at most \( \frac{\delta}{2} \). Hence, the total probability of not having two different balls from any new urn is at most \( \delta \), which proves our claim.

We have seen that even a small sample can portray the \( \varepsilon \)-bad dependencies of a relation quite well. If we then also want to detect \( \varepsilon \)-good dependencies that do not hold in \( r \) and thus come up with a solution for the exact dependency inference problem, we can proceed as follows.

Select the sample \( s \) and construct a cover for \( \text{dep}(s) \). For each dependency \( X \rightarrow Y \) in \( \text{dep}(s) \), check whether it holds in the whole relation \( r \). If \( r \not\models X \rightarrow Y \), add two rows showing this to \( s \). Using the augmented \( s \) as the new sample, infer the dependencies in it again.

The augmentation of the sample is continued until the dependencies of the sample are all true in the original relation. We obtain the following algorithm.
**Algorithm 3.8.** *Input.* A relation $r$ over relation schema $R$.

*Output.* A cover $G$ for the set $\text{dep}(r)$.

*Method.*

\[
s := \text{sample of } r;
\]

\[
F := \text{a cover for } \text{dep}(s);
\]

while $r \not\subseteq F$ do

\[
\text{for each } X \rightarrow Y \in F \text{ such that } r \not\models X \rightarrow Y \text{ do}
\]

\[
\text{add to } s \text{ two rows } u, v \text{ of } r \text{ such that } u[X] = v[X] \text{ and } u[Y] \neq v[Y];
\]

\[
\text{od;}
\]

\[
F := \text{dep}(s);
\]

\[
\text{od;}
\]

output $F$.

**Proposition 3.9.** *On input $r$, Algorithm 3.8 returns a cover for the set $\text{dep}(r)$.*

The choice of a cover $F$ for $\text{dep}(s)$ in Algorithm 3.8 has a twofold effect on the complexity of the algorithm. A large cover consisting of many dependencies leads to a small number of iterations of the while loop, but each execution of the loop is slow. On the other hand, a small cover $F$ may produce more iterations of the while loop, but each iteration is simpler.

### 4. Finding an approximate cover for the set of dependencies

In this section we describe an algorithm for inferring with an arbitrary precision a cover for $\text{dep}(r)$. The algorithm is based on the insightful algorithm of Angluin et al. [3] for learning Horn sentences. Similar applications of their algorithm have been independently done by Dechter and Pearl [9] and Kautz et al. [13].

A Horn clause is a propositional formula of the form $B_1 \land \cdots \land B_k \rightarrow A$, where $B_i$ and $A$ are propositional variables. A Horn sentence is a conjunction of Horn clauses. The algorithm of Angluin et al. [3] uses equivalence and membership queries. That is, the algorithm has access to two oracles: one answers questions of the form "is the formula $H$ equivalent to the Horn sentence $H_*$ to be learned", and the other, "is $x$ a satisfying truth assignment for $H$." The algorithm can exactly identify a Horn sentence in polynomial time using these types of queries.

The first step in adapting this algorithm for dependency inference is easy: functional dependencies can be interpreted as Horn clauses [27]. Assume that we are given the set $R$ of attributes, which we also take as the names of the propositional variables. The dependency $f = (B_1 \rightarrow A)$ corresponds to the Horn clause $C_f = (B_1 \land \cdots \land B_k \rightarrow A)$. Then a set $F$ of dependencies defines a Horn sentence $H_F = \bigwedge_{f \in F} C_f$. A Horn sentence can be identified with the set of truth value assignments that satisfy it, and a truth value assignment can be identified with the set of variables that have the value `true`. Using these identifications, the formula $H_F$ interpreted as a subset of $\mathcal{P}(R)$ is exactly the set $\text{CL}(F)$ of $F$-closed subsets of $R$. 
Let now $H_*$ be the Horn sentence that corresponds to $\text{dep}(r)$. If we could answer the membership and equivalence queries for $H_*$, then the algorithm of Angluin et al. would give us in polynomial time a Horn sentence $H$ that is equivalent to $H_*$. Then, if we denote by $F$ the set of dependencies that corresponds to $H$, we would have $\text{CL}(\text{dep}(r)) = \text{CL}(F)$ and, hence, $F$ would be equivalent to $\text{dep}(r)$.

But membership and equivalence oracles are not directly available to us. Actually, the membership queries are not a problem. As we shall soon see, given $r$ and $X \subseteq R$ we can efficiently determine whether $X$ is $\text{dep}(r)$-closed, i.e., whether $X \in \text{CL}(\text{dep}(r))$ holds.

The equivalence queries, however, would in the dependency inference setting correspond to queries of the form “is $F$ equivalent to $\text{dep}(r)$”, and no polynomial-time algorithm is known for this problem. Indeed, Eiter and Gottlob [12] have shown that a special case of this problem is equivalent to several other problems for which no polynomial-time algorithm is known.

Assume, however, that we are willing to accept as an answer any set $F$ with $d_F(F, \text{dep}(r)) \leq \varepsilon$ for some probability measure $P$ on $\mathcal{P}(R)$ and some error parameter $\varepsilon$. It is well known [2] that we can then answer the $j$th equivalence query “is the current hypothesis $H_j$ equivalent to $H_*$” by choosing

$$q_j(\varepsilon, \delta) = \left(\frac{1}{e}(\ln(1/\delta) + j \ln 2)\right)$$

random subsets $X \in \mathcal{P}(R)$ according to $P$. We answer “yes” if none of the $q_j(\varepsilon, \delta)$ subsets are in $H_j \triangle H_*$. We shall see that in the dependency inference problem we can decide this condition efficiently. This is because $H_* = \text{CL}(\text{dep}(r))$ and $H_j = \text{CL}(F)$ for some $F$, and we can in polynomial time decide whether $X \in \text{CL}(F)$ and $X \in \text{CL}(\text{dep}(r))$ hold. With probability $1 - \delta$, we never answer “yes” if $P(H_j \triangle H_*) > \varepsilon$. Hence, with high probability we do not accept $F$ as correct if $d_F(F, \text{dep}(r)) > \varepsilon$. We are free to choose any $P$, as long as we are able to generate random subsets $X \in \mathcal{P}(R)$ polynomial time.

The following simple but crucial lemma shows how we can answer membership queries and the queries replacing equivalence queries. The lemma is essentially given by Demetrovics and Thi [11, 10]. For two rows $u, u' \in r$, their agree set $\text{ag}(u, u')$ is defined by $\text{ag}(u, u') = \{B \in R \mid s[B] = t[B]\}$.

**Lemma 4.1.** Let $u$ and $u'$ be arbitrary rows, and let $r$ be a relation over $R$. Denote $X = \text{ag}(u, u')$. Then $\{u, u'\} \models \text{dep}(r)$ if and only if $X$ is $\text{dep}(r)$-closed if and only if

$$X = \bigcap \{ \text{ag}(t, t') \mid t, t' \in r, X \subseteq \text{ag}(t, t') \} .$$

**Corollary 4.2.** (i) Given a relation $r$ over $R$ and two rows $u$ and $u'$, one can test in time $\text{sort}(r) + O(|r||R|)$ whether $\{u, u'\} \models \text{dep}(r)$.

(ii) Given a relation $r$ over $R$ and a subset $X$ of $R$, one can test in time $\text{sort}(r) + O(|r||R|)$ whether $X$ is $\text{dep}(r)$-closed.
We can now describe the algorithm for approximate dependency inference.

**Algorithm 4.3.** Approximate computation of the functional dependencies holding in a relation.

**Input.** A relation \( r \) over a schema \( R \) and quantities \( \varepsilon \) and \( \delta \).

**Output.** A set \( F \) of functional dependencies such that \( r \models F \) and with probability at least \( 1 - \delta \) we have \( d_F(F, \text{dep}(r)) \leq \varepsilon \).

**Method.**
1. \( \mathcal{L} := \) the empty list;
2. \( F := \emptyset \);
3. \( j := 1 \);
4. **while** true **do**
5. generate \( q_j(\varepsilon, \delta) \) random subsets of \( R \);
6. **if** every set \( X \) generated is \( \text{dep}(r) \)-closed or not \( F \)-closed
7. **then** return \( F \)
8. **else**
9. Let \( X \) be \( F \)-closed and not \( \text{dep}(r) \)-closed;
10. **for all** \( W \in \mathcal{L} \) **do**
11. test if \( W \cap X \) is not \( \text{dep}(r) \)-closed and \( W \cap X \subset W \);
12. **od**;
13. **if** such a \( W \in \mathcal{L} \) exists **then**
14. replace the first one in \( \mathcal{L} \) by \( W \cap X \);
15. **else** add \( X \) as the last element of \( \mathcal{L} \);
16. **fi**;
17. \( F := \{ Y \rightarrow B \mid Y \in \mathcal{L}, B \notin Y, r \models Y \rightarrow B \} \);
18. \( \mathcal{L} := \emptyset \);
19. \( j := j + 1 \);
20. **od**.

Compared to previous algorithms, the ingenious idea of Angluin et al. [3] is the intersection technique. In the dependency inference terminology, when a new set \( X \) is found that is not \( \text{dep}(r) \)-closed, one looks at previous examples of this type from the list \( \mathcal{L} \) and checks if an example can be shortened by computing the intersection. The basic lemma is the following.

We say that a set \( X \in \mathcal{L} \) violates a dependency \( Y \rightarrow B \) if \( Y \subseteq X \), but \( B \notin X \).

**Lemma 4.4.** Let \( H \) be a cover of \( \text{dep}(r) \).

(i) Every element \( X \in \mathcal{L} \) violates a dependency of \( H \).

(ii) No two elements of \( \mathcal{L} \) violate the same dependency of \( H \).

**Theorem 4.5.** Let \( r \) be a relation with \( n \) attributes. Assume a cover \( H \) for \( \text{dep}(r) \) has \( m \) dependencies with one attribute on the right-hand side. Algorithm 4.3 terminates in \( mn \) iterations of the outermost loop, and returns a set \( F \) of functional dependen-
ties such that \( d_p(F, \text{dep}(R)) \leq \varepsilon \) with probability at least \( 1 - \delta \). The algorithm can be implemented to require \( O(m^2 n^2 / \varepsilon + (\ln 1 / \delta) mn / \varepsilon) \) sorts of the relation \( r \) and an additional book-keeping time \( O(m^2 n^3 / \varepsilon + (\ln 1 / \delta) mn^2 / \varepsilon) \).

The proofs of Lemma 4.4 and Theorem 4.5 are given in Appendix B.

To show the power of the algorithm we give a simple corollary.

**Corollary 4.6.** Assume \( r \) has just one key of length 1. Then Algorithm 4.3 finds it in expected number of \( O(\log n) \) iterations of the main loop.

Note that provided we know that \( r \) has exactly one key consisting of one attribute, and no other functional dependencies, the optimal way for finding this key requires \( \Theta(\log n) \) sorts of the relation.

An open problem related to Algorithm 4.3 is the size of the produced dependency set \( F \) (compared to the size of the optimal cover).

A problem with the above algorithm is that it is inefficient in finding dependencies with long left-hand sides. For a dependency \( B_1 \ldots B_k \rightarrow A \) to be found, one first has to guess a set \( X \) containing \( B_1, \ldots, B_k \), and this happens only with probability \( 2^{-k} \). Further, the set has to be \( F \)-closed.

There are some heuristics that can be used to improve the efficiency of the algorithm. First, whenever a subset \( X \) is generated, one can use instead of \( X \) its closure under \( F \). Second, instead of using the uniform distribution of subsets, one can generate subsets by choosing a pair of rows from \( r \), computing their agree set, and removing one attribute from it. A further variant of the algorithm is obtained by generating a subset \( X \), computing its closure with respect to \( F \) and \( \text{dep}(r) \), and comparing these for equality.

This gives an algorithm that with probability at least \( 1 - \delta \) produces a dependency set \( F \) with \( d_p'(F, \text{dep}(r)) \leq \varepsilon \).

**5. Concluding remarks**

We have considered the problem of inferring the functional dependencies that approximately hold in a given relation. First, we have shown that under three measures of dependency satisfaction in a relation, small samples are sufficient to detect clearly erroneous dependencies. Second, we have demonstrated a randomized output-polynomial algorithm for computing with any accuracy and confidence an approximate cover for the set of functional dependencies that hold in a given relation.

Our results show that approximate techniques can achieve good results in the dependency inference problem. Practical experiments are needed to verify what the properties of real relations are, and also what is the best way of combining approximate dependency inference and verification of the dependencies obtained using it.

There are several interesting directions for extending the results about using samples to detect clearly erroneous dependencies. We have recently generalized them to checking approximately not only functional dependencies but also all properties of relations.
that can be expressed by universal sentences in tuple relational calculus [14]. Another
direction, which we are currently studying, is to consider alternative error measures and
try to relate the required sample sizes to other properties of the measures. We have
also considered finding all the functional dependencies that are \( \varepsilon \)-good in a relation.
This is possible by a straightforward modification of one of the algorithms [21] for
finding all functional dependencies that hold exactly.

The techniques we have used are fairly independent of the actual class of depen-
dencies used. Thus, they can be generalized to more general constraints, as long as
the constraints can be formulated as Horn clauses. It is an interesting open ques-
tion whether similar randomized algorithms can be obtained for the problems related
to dependency inference described by Eiter and Gottlob [12]. On the other hand,
it would be interesting to see what can be achieved by a careful application of
the intersection technique of Angluin et al. [3] to the dependency inference prob-
lem. There is always the possibility of an exact output-polynomial algorithm for the
problem.

Beeri et al. [5] describe some preliminary results about the dependencies that hold
in a random relation. The following general question is to our knowledge open: given
a random relation satisfying a dependency set \( F \), what other dependencies does the
relation satisfy?

Acknowledgements

The work of Jyrki Kivinen has been partially funded by the Academy of Finland and
by Emil Aaltonen Foundation, and part of it was done while he was visiting University
of California at Santa Cruz. The work of Heikki Mannila has been partially funded by
the Academy of Finland, and part of it was done while he was visiting the Technical
University of Vienna.

Appendix A. Properties of error measures

It is relatively straightforward to prove the following relationships between the mea-
sures.

**Proposition A.1.** Let \( r \) be a relation over \( R \), and let \( X \) and \( Y \) be two sets of attributes.
For \( i = 1, 2, 3 \), let \( g_i = g_i(X \rightarrow Y, r) \). If the dependency \( X \rightarrow Y \) holds in \( r \), then \( g_i = 0 \)
for all \( i \). Otherwise, the following inequalities hold:

\[
g_2/|r| \leq g_1 \leq g_2^2 - g_2/|r|,
\]

\[
2g_3/|r| \leq g_1 \leq 2g_3 - g_3^2 - g_3/|r|,
\]

\[
g_3 + 1/|r| \leq g_2 \leq 1.
\]

These bounds are tight.
Proof. The case with \( r \models X \rightarrow Y \) is clear. Assume that \( X \rightarrow Y \) does not hold in \( r \). For \( i = 1, 2, 3 \), let \( G_i = G_i(X \rightarrow Y, r) \). We prove bounds for the values \( G_i \). The bounds for the values \( g_i \) follow directly. Let the set of violating pairs be

\[
P = \{ (u, v) \in r^2 \mid (u, v) \text{ violates } X \rightarrow Y \}
\]

and the set of violating tuples

\[
Q = \{ u \in r \mid (u, v) \text{ violates } X \rightarrow Y \text{ for some } v \in r \}.
\]

Then \( G_1 = |P| \) and \( G_2 = |Q| \).

Since \( P \subseteq Q \times Q \) and \( u \neq v \) for all \((u, v) \in P\), we get \(|P| \leq |Q|(|Q| - 1)\). Equality holds if all the violating tuples \( u \in Q \) have the same value \( u[X] \) but different values \( u[Y] \).

On the other hand, \( Q \) is the set of tuples that appear as a first component in some pair in \( P \), so we must have \(|Q| \leq |P|\). Here equality holds if for each violating tuple \( u \in Q \) there is exactly one tuple \( v \in Q \) such that \((u, v) \in P\). Thus, we have the tight bounds \( G_2 \leq G_1 \leq G_2^2 - G_2 \).

To prove an upper bound for \( G_3 \) in terms of \( G_1 \), let \( \leq \) be an arbitrary linear order on the tuples of \( r \), and

\[
P' = \{ (u, v) \in P \mid u < v \}.
\]

Since \((u, v) \in P \) if and only if \((v, u) \in P\), we have \(|P'| = |P|/2\). Now construct a subrelation \( s \subseteq r \) by removing from \( r \) every tuple \( u \) such that \((u, v) \in P' \) for some \( v \). Then \( s \models X \rightarrow Y \), so \( G_3 \leq |r| - |s| = G_1/2 \). Equality holds if \( u \in Q \) has exactly one other violating tuple \( v \in Q \) such that \((u, v) \in P\).

On the other hand, let \( s \subseteq r \) be a subrelation of maximum size such that \( s \models X \rightarrow Y \).

Then \(|s| = |r| - G_3\). Let \( r[X] = \{ x_1, \ldots, x_k \} \) be the set of values \( u[X] \) taken by the attribute set \( X \) on the tuples \( u \in r \). For \( i = 1, \ldots, k \), let

\[
A_i = \{ u \in s \mid u[X] = x_i \}
\]

and

\[
B_i = \{ u \in r - s \mid u[X] = x_i \}.
\]

Let \( n_i = |A_i| \) and \( m_i = |B_i| \). Then \( \sum_i n_i = |r| - G_3 \) and \( \sum_i m_i = G_3 \). If a pair \((u, v)\) violates \( X \rightarrow Y \), it satisfies for exactly one \( i \) exactly one of the following conditions:

1. \( u \in A_i \) and \( v \in B_i \),
2. \( u \in B_i \) and \( v \in A_i \) or
3. \( u \in B_i \) and \( v \in B_i - \{ u \} \).

Therefore,

\[
G_1 \leq \sum_{i=1}^k (2n_i m_i + m_i m_i - 1))
\]

\[
\leq 2 \left( \sum_{i=1}^k n_i \right) \left( \sum_{i=1}^k m_i \right) + \left( \sum_{i=1}^k m_i \right)^2 - \sum_{i=1}^k m_i
\]
\[ = 2 (|r| - G_2) G_3 + G_3^2 - G_3 \]
\[ = 2 |r| G_3 - G_3^2 - G_3. \]

Equality holds if all the tuples \( u \in r \) have the same value \( u[X] \) and, additionally, all the tuples \( u \in r - s \) have a different value \( u[Y] \).

To see the relationship between \( G_2 \) and \( G_3 \), construct a subrelation \( s \subseteq r \) by removing from \( r \) all but one of the violating tuples \( v \in Q \). Then \( G_2 = |r| - |s| + 1 \). From the tuples remaining in \( s \) we clearly cannot construct a violating pair. Hence, \( s \models X \rightarrow Y \), and we have \( G_3 \leq |r| - |s| = G_2 - 1 \). This bound is tight if all the violating tuples \( u \in Q \) have the same value \( u[X] \) but a different value \( u[Y] \).

On the other hand, trivially \( G_2 \leq |r| \). If all the tuples \( u \in r \) have the same value \( u[X] \) and all but one of them have the same value \( u[Y] \), we have \( G_3 = 1 \) but \( G_2 = |r| \).

For each measure one can form an alternative one by considering not \( r \) but only the projected relation \( r[XY] \).

**Example A.2.** Consider the relation \( r \) over \( ABC \) that contains the tuple \( (|r|, 1, 0) \) and for \( j = 1, \ldots, |r| - 1 \) the tuple \( (j, 1, 1) \). We have \( g_1(B \rightarrow C, r) = 2(|r| - 1)/|r|^2 \), and hence \( r \) almost satisfies \( B \rightarrow C \). If we consider the variant where projection to \( BC \) is computed first, we get \( g_1(B \rightarrow C, r[BC]) = 1/2 \), which is the highest possible since \( |r[BC]| = 2 \).

One of the useful properties of functional dependencies is the existence of a simple, sound and complete axiomatization for them. For example, the following axiomatization can be used: (1) \( X \rightarrow X' \) for all \( X' \subseteq X \); (2) if \( X \rightarrow Y \) and \( Y \rightarrow Z \), then \( X \rightarrow Z \); and (3) if \( X \rightarrow Y \) and \( X \rightarrow Z \), then \( X \rightarrowYZ \).

Axiom (1) implies that if \( X' \subseteq X \), then \( g_i(X \rightarrow X', r) = 0 \) for all relations \( r \) and for all our error measures \( g_i \). We might also wish to apply Axioms (2) and (3) in order to derive new dependencies from dependencies we have somehow found to have small error. The following results show that if we do so, the error of the derived dependencies does not exceed the sum of the errors of the dependencies used as premises.

**Proposition A.3.** For all the measures \( g_i \) where \( i = 1, 2, 3 \), all sets \( X, Y \) and \( Z \) of attributes and all relations \( r \), the following inequalities hold:

\[ g_i(X \rightarrow Z, r) \leq g_i(X \rightarrow Y, r) + g_i(Y \rightarrow Z, r). \]  
\[ g_i(X \rightarrow YZ, r) \leq g_i(X \rightarrow Y, r) + g_i(X \rightarrow Z, r). \]  

**Proof.** Consider first the measure \( g_1 \). If a pair \((u, v)\) of tuples in the relation \( r \) violates the dependency \( X \rightarrow Z \), then \( u[X] = v[X] \) but \( u[Z] \neq v[Z] \). If then \( u[Y] = v[Y] \), the pair violates the dependency \( Y \rightarrow Z \). Otherwise the pair violates the dependency \( X \rightarrow Y \). Therefore, we have \( G_1(X \rightarrow Z, r) \leq G_1(X \rightarrow Y, r) + G_1(Y \rightarrow Z, r) \), and (A.1) follows. Similarly, if a pair \((u, v)\) violates the dependency \( X \rightarrow YZ \), we must have...
\[ u[X] = v[X] \text{ but } u[Y] \neq v[Y] \text{ or } u[Z] \neq v[Z], \] so the pair violates the dependency \( X \rightarrow Y \) or the dependency \( X \rightarrow Z \). The same arguments show (A.1) and (A.2) to hold for the measure \( g_2 \).

Consider now the measure \( g_3 \). Let \( r_1 \subseteq r \) and \( r_2 \subseteq r \) be subrelations of \( r \) such that \( r_1 \models X \rightarrow Y \) and \( r_2 \models Y \rightarrow Z \). If we then define \( s = r_1 \cap r_2 \) to consist of the tuples that are in both \( r_1 \) and \( r_2 \), we have \( s \models X \rightarrow Z \). This implies \( \left| r - r_1 \right| = m_1 \) and \( \left| r - r_2 \right| = m_2 \), then \( \left| r - s \right| \leq m_1 + m_2 \). Hence, we have \( G_3(X \rightarrow Z, r) \leq G_3(X \rightarrow Y, r) + G_3(Y \rightarrow Z, r) \), and (A.1) for the measure \( g_3 \) follows. The same argument also gives (A.2).

The following example show that the estimates (A.1) and (A.2) cannot be improved in general.

**Example A.4.** Let \( p \geq 2 \) and \( q \geq 2 \) be integers and \( m \geq p+q \). We first consider inferring \( A \rightarrow C \) from \( A \rightarrow B \) and \( B \rightarrow C \). Let the relation \( r \) over \( ABC \) contain \( m \) tuples as follows: for \( i = 1, \ldots, p \), the tuple \((1, i, i)\); for \( i = 1, \ldots, q \), the tuple \((0, 0, i)\); and for \( i = p+q+1, \ldots, m \), the tuple \((i, i, i)\). Then \( G_1(A \rightarrow B, r) = p(p-1) \), \( G_1(B \rightarrow C, r) = q(q-1) \) and \( G_1(A \rightarrow C, r) = p(p-1) + q(q-1) \). We also have \( G_2(A \rightarrow B, r) = p \), \( G_2(B \rightarrow C, r) = q \) and \( G_2(A \rightarrow C, r) = p+q \). Finally, we have \( G_3(A \rightarrow B, r) = p-1 \), \( G_3(A \rightarrow C, r) = q-1 \) and \( G_3(A \rightarrow BC, r) = p+q-2 \).

Consider now inferring \( A \rightarrow BC \) from \( A \rightarrow B \) and \( A \rightarrow C \). Again, let \( p \geq 2 \) and \( q \geq 2 \) be integers and \( m \geq p+q \). Let the relation \( r \) over \( ABC \) contain \( m \) tuples as follows: for \( i = 1, \ldots, p \), the tuple \((1, i, 0)\); for \( i = 1, \ldots, q \), the tuple \((0, 0, i)\); and for \( i = p+q+1, \ldots, m \), the tuple \((i, i, i)\). Then \( G_1(A \rightarrow B, r) = p(p-1) \), \( G_1(A \rightarrow C, r) = q(q-1) \) and \( G_1(A \rightarrow BC, r) = p(p-1) + q(q-1) \). We also have \( G_2(A \rightarrow B, r) = p \), \( G_2(A \rightarrow C, r) = q \) and \( G_2(A \rightarrow BC, r) = p+q \). Finally, we have \( G_3(A \rightarrow B, r) = p-1 \), \( G_3(A \rightarrow C, r) = q-1 \) and \( G_3(A \rightarrow BC, r) = p+q-2 \). □

**Appendix B. Proof of Lemma 4.4 and Theorem 4.5**

The proof of the correctness of Algorithm 4.3 is basically the same as the proof given by Angluin et al. [3] for their algorithm; our somewhat different setting makes some modifications necessary. We need some preliminary lemmas.

**Lemma B.1.** Let \( H \) be a cover of \( \text{dep}(r) \), and let \( Z \rightarrow B \in H \). Assume that during the execution of Algorithm 4.3 the list \( \mathcal{L} \) contains an element \( W \) such that \( Z \subseteq W \), and that the set \( X \) chosen on line 9 of the algorithm violates the dependency \( Z \rightarrow B \). Then some element \( W' \) of \( \mathcal{L} \) preceding or equal to \( W \) will be replaced by \( W' \cap X \) on line 15.

**Proof.** Induction on the number \( k \) of \( F \)-closed but not \( \text{dep}(r) \)-closed sets \( X \) found in the execution. If \( k = 0 \), the list is empty and the claim holds. Assume \( k > 1 \). If some element of \( \mathcal{L} \) preceding \( W \) is replaced, the claim holds. If not, we have to show that
$W$ will be replaced by $W \cap X$ on line 15. Our assumptions imply that $W \cap X$ is not $\text{dep}(r)$-closed. Hence, it remains to show that $W \cap X \subseteq W$.

If $W \cap X = W$, then $W \subseteq X$ and $B \notin W$. Consider the dependency $W \rightarrow B$. As $Z \rightarrow B \in H$, and the set $H$ is a cover of $\text{dep}(r)$, and $Z \subseteq W$, we have $W \rightarrow B \in \text{dep}(r)$. Since $W$ is in the list $\mathcal{L}$, this implies $W \rightarrow B \in F$. But $W \subseteq X$ and $B \notin X$, contradicting the assumption that $X$ is $F$-closed. □

**Lemma B.2.** Let the sets $W_1$ and $W_2$ occur in the list $\mathcal{L}$, and assume $W_1$ occurs before $W_2$. Let $H$ be a cover of $\text{dep}(r)$, and let $Z \rightarrow B \in H$. If $W_2$ violates $Z \rightarrow B$, then $Z \subseteq W_1$. Specifically, if $W_2$ violates $Z \rightarrow B$, then $W_1$ does not violate $Z \rightarrow B$.

**Proof.** Induction on the number of iterations of the loop on lines 4–21. The basic case is clear, since $\mathcal{L}$ is empty.

Assume the induction hypothesis, and suppose for the sake of contradiction that a new update of the list $\mathcal{L}$ causes the claim to fail. There are two cases, the first being where a new set $X$ is inserted at the end of the list. By the induction hypothesis, the only possibility for the failure of the claim is that $X$ violates $Z \rightarrow B$ and some $W$ already in the list $\mathcal{L}$ contains $Z$. But this contradicts Lemma B.1.

In the second case the list $\mathcal{L}$ is updated by replacing a set $W$ by $W \cap X$. There are two subcases. In the first the set $W_1$ contains $Z$ and $W_2 = W \cap X$ occurs later in $\mathcal{L}$. Since $W_2$ violates $Z \rightarrow B$, either $W$ violates $Z \rightarrow B$ or $X$ violates $Z \rightarrow B$. If $X$ violates $Z \rightarrow B$, then again by Lemma B.1 some element preceding or equal to $W_1$ should have been replaced, and not $W_2$. The assumption that $W$ violates $Z \rightarrow B$ contradicts the induction assumption. The second subcase is that $Z \subseteq W_1$, where $W_1 = W \cap X$ for some $W$ that occurred before $W_2$ in the list. This would imply $Z \subseteq W$, contrary to the induction hypothesis. □

**Proof of Lemma 4.4.** (i) Each element added to $\mathcal{L}$ (either at the end or due to a replacement) is not $\text{dep}(r)$-closed, i.e., violates some dependency of every cover of $\text{dep}(r)$.

(ii) Special case of Lemma B.2. □

**Proof of Theorem 4.5.** We first consider for a fixed $j$ the probability of the event that at the $j$th iteration of the main loop, the algorithm outputs a dependency set $F$ such that $d_p(F, \text{dep}(r)) > \varepsilon$. Let $m = q_j(\varepsilon, \delta)$. During the execution of the algorithm, the dependency set $F$ satisfies $F \subseteq \text{dep}(r)$, since only dependencies that hold in $r$ are added into $F$. Hence, $\text{CL}(\text{dep}(r)) \subseteq \text{CL}(F)$, so $d_p(F, \text{dep}(r)) = P(\text{CL}(F) - \text{CL}(\text{dep}(r)))$. Assume now that at the beginning of the $j$th iteration, the dependency set $F$ satisfies $d_p(F, \text{dep}(r)) > \varepsilon$. Then a random set $X \subseteq R$ is $F$-closed but not $\text{dep}(r)$-closed with probability greater than $\varepsilon$. If the algorithm outputs $F$, it has at $m$ independent trials drawn $X$ from a set that has probability less than $1 - \varepsilon$. The probability of this event is less than

$$(1 - \varepsilon)^m \leq e^{-mc} \leq \delta/2^j.$$
Since this holds for all \( j \), the probability that at some iteration \( j \) the algorithm outputs a dependency set \( F \) such that \( d_p(F, \text{dep}(r)) > \varepsilon \) holds is at most
\[
\sum_{j=1}^{\infty} \frac{\delta}{2^j} = \delta.
\]

We now turn to the run time of the algorithm. Each iteration of the main loop either adds a new set to the list \( \mathcal{L} \), or refines an existing element of the list. By the above lemmas, the list can contain at most \( m \) members, and each member can have at most \( n \) elements. Thus, the maximum number of changes to \( \mathcal{L} \) is \( mn \); this is also an upper bound for the number of iterations of the main loop. Iteration \( j \) contains at most
- \( q_j(\varepsilon, \delta) \) generations of a random subset \( X \),
- on line 6, \( q_j(\varepsilon, \delta) \) checks for \( \text{dep}(r) \)-closedness and \( F \)-closedness,
- on lines 10–13, \( mn \) checks for \( \text{dep}(r) \)-closedness and of the condition \( W \cap X \subseteq W \) and
- on line 18, \( mn \) checks of the condition \( r \models Y \rightarrow B \).

By Lemma 4.2 we know that \( \text{dep}(r) \)-closedness of \( X \), as well as the condition \( r \models Y \rightarrow B \), can be checked in time \( O(\text{sort}(r)) \). By using the bound \( q_j(\varepsilon, \delta) < (1/\varepsilon)(\ln(1/\delta) + mn \ln 2) \) we see that the total number of sorts of the relation is \( O(m^2 n^2 / \varepsilon + (\ln 1/\delta)mn/\varepsilon) \).

It is known that given \( X \subseteq R \) and a set \( G \) of functional dependencies over \( R \), the time needed to decide whether \( X \) is \( G \)-closed is linear in the total length of the dependencies in \( G \) [30, 20]. For our algorithm, we can compress \( F \) by combining the dependencies with a common left-hand side. That is, we can represent \( F \) by
\[
G = \{ Y \rightarrow Y' \mid Y \in \mathcal{L}, Y' = \{ B \notin Y, r \models Y \rightarrow B \} \}.
\]
The total length of the dependencies in \( G \) is \( O(mn^2) \). Since \( F \) and \( G \) are equivalent, we can test for \( F \)-closedness of \( X \) in time \( O(mn^2) \). Hence, the total book-keeping time for our algorithm is \( O(m^2 n^2 / \varepsilon + (\ln 1/\delta)mn^2 / \varepsilon) \). \( \square \)

References