DISCOVERING ONTOLOGY FUNCTIONAL DEPENDENCIES

DISCOVERING ONTOLOGY FUNCTIONAL DEPENDENCIES

BY

SRIDEVI BASKARAN, B.E.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2016) McMaster University

(Computer Science) Hamilton, Ontario, Canada

TITLE: Discovering Ontology Functional Dependencies

AUTHOR: Sridevi Baskaran, B.E.

SUPERVISOR: Dr.Fei Chiang

Dedicated to my family.

Abstract

Functional Dependencies (FDs) are commonly used in data cleaning to identify dirty and inconsistent data values. However, many errors require user input for specific domain knowledge. For example, let us consider the drugs, *Advil* and *Crocin*. FDs will consider these two drugs different because they are not syntactically equal. However, *Advil* and *Crocin* are synonyms as they are two different drugs with similar chemical compounds but marketed under distinct names in different countries. While FDs have traditionally been used in existing data cleaning solutions to model syntactic equivalence, they are not able to model broader relationships (e.g., synonym, Is-A (Inheritance)) defined by ontologies.

In this thesis, we take a first step to discover a new dependency called *Ontology Functional Dependencies (OFDs)*. OFDs model attribute relationships based on relationships in a given ontology. We present two effective algorithms to discover OFDs using synonyms and inheritance relationships. Our discovery algorithms search for minimal OFDs and prune the redundant ones. Both algorithms traverse the search lattice in a level-wise Breadth First Search (BFS) manner. In addition, we have developed a set of pruning rules so that we can avoid considering unnecessary candidates in the search lattice. We present an experimental study describing the performance

and scalability of our techniques. Experimental results show that both algorithms are effective in practice and discover OFDs efficiently for large datasets with millions of tuples. We also present a qualitative study showing that the discovered OFDs are meaningful with high precision and recall.

Acknowledgement

I am deeply indebted to Dr. Fei Chiang, my revered Supervisor for her esteemed guidance and consistent help.

I am profusely thankful to Dr. Jaroslaw Szlichta for his suggestions and encouragement.

Contents

List of Figures

Chapter 1

Introduction

Achieving high data quality is one of the most significant problems in data management. For organizations to be better served by their information systems, a high degree of data quality is required. However, real-life data are often dirty: inconsistent, duplicated, inaccurate, incomplete, or stale. Dirty data in a database generate misleading or biased analytical results and decisions, and lead to loss of revenues, credibility and customers. Data quality management includes detection and correction of syntactic and semantic errors in the data and adds value to business processes.

Relation r contains a set of attributes A_1, A_2 ... A_n. For example, Table 1.1 has ten rows and five attributes namely *Countrycode*, *Country*, *Symptoms*, *Diagnosis* and *Medicine*. A Functional Dependency (FD) over r can be represented as $F : A \rightarrow B$, where A and B are attribute sets in r. A is referred to as the antecedent while B is referred to as the consequent of F. We say $r \models F$ if and only if for every pair of tuples t_1 and t_2, where t_1, $t_2 \in r$, $t_1[\mathrm{A}] = t_2[\mathrm{A}]$ implies that $t_1[\mathrm{B}] = t_2[\mathrm{B}]$.

Tuple	Countrycode	Country	Symptoms	Diagnosis	Medicine
t_1	US	United States	Stomach ache	Ulcer	Pepcid
t_2	CA	Canada	Stomach ache	Ulcer	Zantac
t_3	AU	Australia	Stomach ache	Ulcer	Histamine
t_4	US	America	Head ache	Migrane	Advil
t_5	IN	India	Head ache	Migrane	Paracetamol
t_6	CA	Canada	Head ache	Migrane	Acetaminophen
t_7	IN	Bharat	Stroke	Hypertension	Methadone
t_8	IN	India	Stroke	Hypertension	Morphine
t_9	US	USA	Stroke	Hypertension	Opioid
t_{10}	CA	Canada	Stroke	Hypertension	Morphine

Table 1.1: Records from 2008 clinical trials dataset [16]

While FDs are able to capture attribute relationships based on equality, they are unable to identify attributes that may be synonymous or syntactically equivalent. Hence we propose Ontology Functional Dependencies (OFDs).

Our work of discovering Ontology Functional Dependencies to clean the data based on the notion of equivalence property, in which the concepts are considered to be equivalent if they are identical or if they represent the same entity. For example, United States of America can be represented as USA, America or US. These concepts are considered to be equivalent as they represent the same country. Consequently, our work finds similarities with two areas of research: dependency discovery and data cleaning.

1.1 Dependency Discovery

Algorithms TANE [4] and Fun [26] model the search lattice as a set of attribute combinations for traversing it. All these algorithms follow the Breadth First strategy for lattice traversal. Breadth first approach works level-wise, in which there are k levels, wherein k ranges from 1 ... n, where n is the number of attributes in the relation. The results of each level are used to generate the candidates for the next $k+1$ level. In the worst case, this approach leads to exponential run-time behavior based on the number of attributes in the relation.

The TANE algorithm is based on three main concepts [4]:
(a) Partition refinement to check if a functional dependency holds
(b) Leve-wise lattice traversal to generate FD candidates
(c) Pruning rules to dynamically reduce the search space. TANE's search space pruning is based on the fact that for a complete result only minimal functional dependencies need be discovered.

Algorithm Dep-Miner follows a different approach than the traditional Breadth First Search approach. Dep-Miner models the search space as a cross product of all tuples. They search for attributes that agree on values in certian tuple pairs. In addition, Dep-Miner calculates covers called agree sets to determine the dependencies in the relation. They use the agree sets to derive all FDs in the relation. Computation of agree sets lead to quadratic time complexity based on number of tuples in the relation.

In our work, we use the same approach used by TANE to model the search space

as a set of attribute combinations and use level-wise BFS strategy for the traversal. In addition, we adapt optimisation techniques to avoid visiting redundant candidates, thereby improving the efficiency of our algorithms.

1.2 Data Cleaning

There is a large amount of work related to data cleaning using Functional Dependencies [1] [9] [10] [27]. Bohannon et al [1] use equality classes to group the tuples for which updates will be suggested to correct the error values. Their research used a cost based technique for repairing the data. Furthermore, they present many heuristics for manipulating every equality class based on the cost of adding tuples to the class.

Chiang et al [27] focus on efficient usage of FDs to improve the quality of data. They propose a model that uses both data and existing constraints for data cleaning. Their data repair algorithm search for data modifications such that existing constraints hold on the relation. Their constraint repair algorithm search for additional attributes in the relation that can be added to the existing constraints to resolve the inconsistencies in the database. Addition of new attributes to the existing constraints is based on the variance of information in the database.

Cong et al [3] propose a new algorithm for finding data repairs based on Conditional Functional Dependencies (CFDs). CFDs are dependencies that hold conditionally over a portion of data as opposed to FDs that hold over the entire dataset. They focus on minimal changes to the dirty database D' based on discovered CFDs

such that it closely matches to the original database D and also satisfies certain constraints. In addition, they propose a statistical method for finding the repair model based on the user defined accuracy rate.

All these algorithms use FDs or CFDs for cleaning the data. They are unable to identify attribute relationships that are semantically similar. Hence they cannot be used to clean the data based on the notion of equivalence property. In our work, we discover a new class of dependencies called Ontology Functional Dependencies to clean the data based on attributes that may be both semantically or syntactically equivalent.

1.3 Ontology Functional Dependency Discovery

All the current algorithms focus on cleaning the data using FDs. They do not use ontologies for finding the semantic equivalence among data values. Two concepts are said to be equivalent if they are identical or represent the same entity.

Ontologies are the manifestation of shared conceptualization of a domain. Domain specific knowledge that is needed to validate and clean the data is represented using ontologies. Usually, domain specific knowledge is the combination of data, its ontology and the relationships among them. Ontology represents the concepts and different relationships between them. Concepts in the ontologies are represented as ovals and relationships between them are represented using arrows. In our work, we have used ontologies for storing the synonyms and inheritance relationships among

concepts. In our research study, we discover a new flavor of dependency called Ontology Functional Dependencies (OFDs) to capture the constraints based on synonyms and inheritance relationships in the databases.

For example, *Table 1.1* shows a sample of clinical trial records containing patient country codes, country, symptoms, diagnosis and medicine. Let us consider the FD: $[Countrycode] \rightarrow [Country]$, which means if the attribute *countrycode* is equal to a set of tuples, then their corresponding attribute *country* should also be equal. Tuples t_1, t_4 and t_9 do not satisfy this FD as United States, America, and USA are not syntactically equivalent and similarly, the tuples t_5, t_7 and t_8 violate this FD. However, we know that the United States is synonymous with America and USA, and t_1, t_4 and t_9 all refer to the same country. Similarly, Bharat in t_7 is also synonymous with India as 'India' is popularly called 'Bharat'. Hence we have a synonym relationship present in the relation that can be defined using the domain knowledge in the ontologies.

Let us consider another FD $[Symptoms, Diagnosis] \rightarrow [medicine]$. Tuples t_1, t_2, t_3 and t_4, t_5, t_6 and t_7,t_8,t_9, t_{10} do not satisfy the dependency as the consequent values all refer to different medications. However, from the medical ontology (Figure 1.1), we see that the values participate in an inheritance relationship. Both 'Pepcid' and 'Zantac' are 'Histamine', both Morphine and 'Methadone' are 'Opioid', and both 'Histamine' and 'Opioid' in turn are 'Analgesic'. Hence we have an inheritance relationship that is defined among different concepts in the relation that is defined by ontologies.

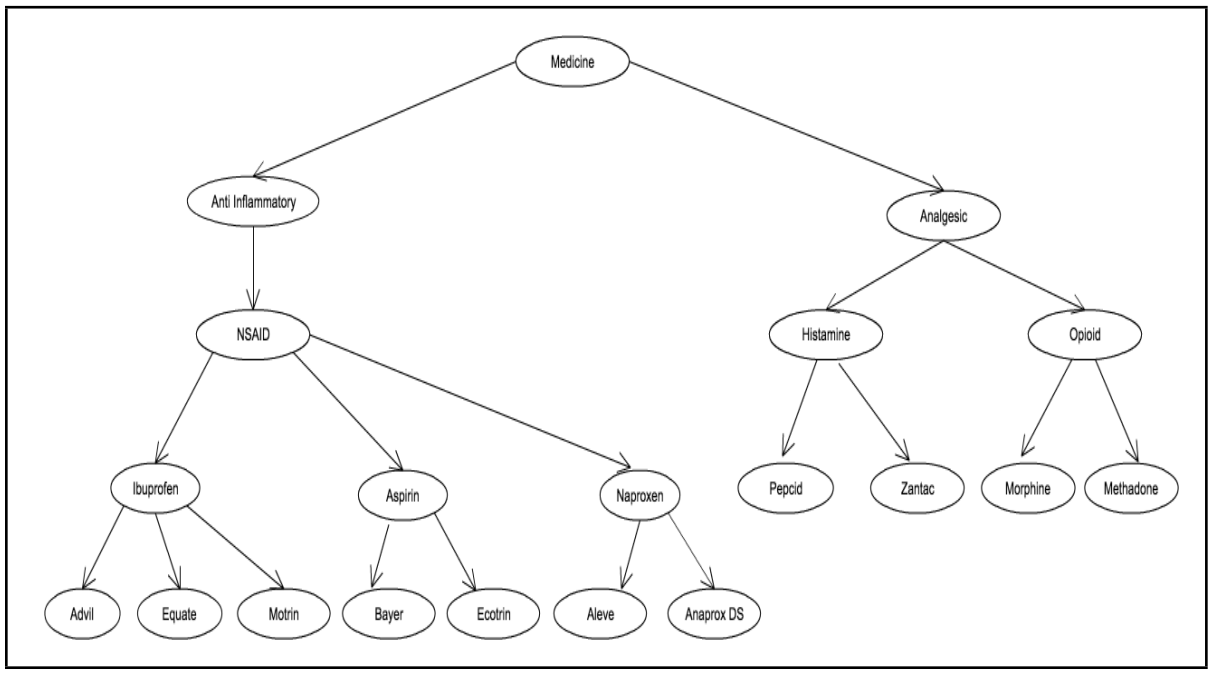


Figure 1.1: **Medical Ontology [18]**

In the present study, we take a first step by defining a new class of dependencies called OFDs that capture the relationships defined in an ontology. Specifically, we focus on the synonym and is-a (inheritance) relationships between two attribute values. We make the following contributions:

1. **Algorithms that efficiently and effectively discover OFDs in the datasets.** We define OFDs for discovering the synonym and inheritance relationships that are defined by an ontology. Our discovery algorithms identify the OFDs over a relation.

2. **Optimisation techniques that improve the performance of our algorithms.** We describe a set of optimisation techniques that prune the search space and improve the running time of the algorithms.

3. **An experimental evaluation showing the performance and effectiveness of our techniques using real data.** We experiment with two different real datasets that span up to 1M records. We also evaluate the performance and scalability of our techniques.

Chapter 2

Background

2.1 Relational Model

We assume that the data is stored in tables known as relations. A single row in the relation represents a single tuple (or record), and each tuple can be described by a set of columns known as attributes. For some tuple, each attribute can only be associated with a single value from a set of allowed values, known as the attribute domain. The set of tuples in a relation can be denoted by r and the total number of tuples is given by the cardinality of r, denoted by $N = |r|$. One example of a relation is shown in Table 2.2. This relation contains five tuples and five attributes: tuple, FName, LName, City and Country.

Many popular database management systems follow the relational model and are ubiquitous in industrial applications. Conceptually, relations are easy to understand and visualize. Often, business owners define rules over relations in an endeavor to

Tuple	FName	LName	City	Country
t_1	Alice	Barry	Toronto	Canada
t_2	John	Smith	Vancouver	Canada
t_3	Nisha	Dominic	Melbourne	Australia
t_4	Bill	Peter	Calgary	Canada
t_5	Alex	Burns	Adelaide	United States

Table 2.2: Relational table

control the quality of the data that is stored in the relations. Incoming data that violate these rules need to be examined carefully. These rules are known as constraints, and a variety of constraints exist.

2.2 Ontologies

In this section, we present ontologies as an important source of semantic knowledge that can be useful for a wide range of applications in database. First, we define what an ontology is, illustrate its characteristics, and explain its role for data management.

2.2.1 Definition of Ontology

Ontologies are stated as "explicit specifications of conceptualizations of a domain" [13]. In our work, we consider an ontology as a hierarchy of concepts with relationships between them. We consider the synonyms and inheritance relationships between concepts in the ontology.

2.2.2 Description of Real World Objects

Real world objects are described in ontologies using concepts and relationships as follows:

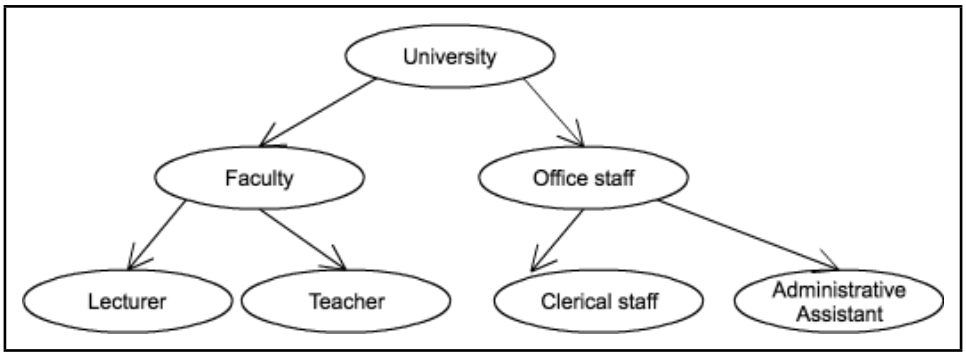


Figure 2.2: **Example Ontology**

- **Concepts** can be defined as abstractions that denote real-world objects having common properties. They are usually referred by their titles, which are words taken from some natural language. These words must be unambiguous. Furthermore, concepts are organized in a taxonomy, which relates more specialized concepts (sub-concepts) to more generalized concepts (super-concepts) that subsume them. Sub-concepts related to the same super-concept are disjunctive.In Figure 2.2, concepts are defined in ovals. Some of the concepts include *university*, *faculty*, *staff*. Relationships are represented by arrows.

- **Relationships** define dependencies between entities of the same concept and entities of other concepts. There are two classes of domain based relationships: generic relationships and domain-specific relationships. The first class includes

specialization (ISA), synonymy (SynOf), and decomposition (PartOf) relationships. The second class contains relationships that are specific to the domain of ontology. In Figure 2.2, concepts *Lecturer* and *Teacher* are synonyms as they both represent the same category of faculty members but represented in different titles in different countries or educational institutions. Concepts *Lecturer* and *Teacher* are a type-of *faculty members*, through which we define the inheritance relationship.

Chapter 3

Preliminaries

A *Functional Dependency* (*FD*) F over a relation r is represented as $X \rightarrow Y$, where X, Y are attribute sets in r. We will denote the attributes of F by $F = XY$. We will use the short form $|F|$ to denote $\mid XY \mid$, i.e. the cardinality of XY. Relation r satisfies F (written as $r \models F$) if for every pair of tuples t_1, t_2 in r, if t_1[X] = t_2[X], then t_1[Y] = t_2[Y].

We introduce the notations that are used throughout this paper in Table 3.3.

Table 3.3: Notational conventions

- A small letter represents a specific relation in R: r.
- Capital letter S represents the ontology: S.
- Capital letter near the end of the alphabet represent a set of attributes: X, Y and Z.

- Capital letters near the beginning of the alphabet represent single attributes: A and B.

In the search for OFDs, we assume an ontology S that contains a set of concepts C. For example, in Figure 1.1, all classes are represented in ovals. Partition of X, Π_X, is defined as a set of equivalence classes that have tuples with equal values in the attribute list X. Let n be the number of attributes in R. Let x_i represent an equivalence class with a representative i that is equal to the smallest tuple id in the class and $|x_i|$ be the size of the class. For example, in Table 1.1, $\Pi_{Countrycode}$ = {{1,4,9}{2,6,10}{3}{5,7,8}}. Similarly, we can compute the partitions for other attributes in the relation.

Given a relational instance r, our objective is to discover a minimal set of OFDs. Classes C contain all possible senses for a given attribute value. For example, under the sense of automobiles, *Jaguar* is a car whereas under the context of animals, *Jaguar* is a Tiger. Canonical names(C) is the set of all canonical names for a given class. They are unique string representations for each concept in the ontology that occupies less memory and is very efficient when we check for the validity of OFDs. For example, canonical names of medicines in Table 1.1 is as follows: for *Pepcid* is *pd*, for *Zantac* is *zn*, for *Advil* is *ad*. Let $g_x[X] = \{t|$ t $\in$ r and $t[X] = x\}$.

We consider deriving OFDs using synonyms and inheritance relationships. Synonym OFDs are represented as $X \rightarrow_{syn} Y$.

Synonym relationship

A relation r satisfies a *Synonym FD* $X \rightarrow_{syn} Y$, if for each attribute $A \in Y$, for each $x \in \Pi_X(\mathbf{r})$, there exists a class C, such that $\Pi_A(g_x[X]) \subseteq synonym(C)$. Terms are string representations of classes in the ontology defined with predicate Synonym(C). A class with multiple instances, i.e.,|Synonym(C)| > 1, contains alternative string representations for the class (synonyms).

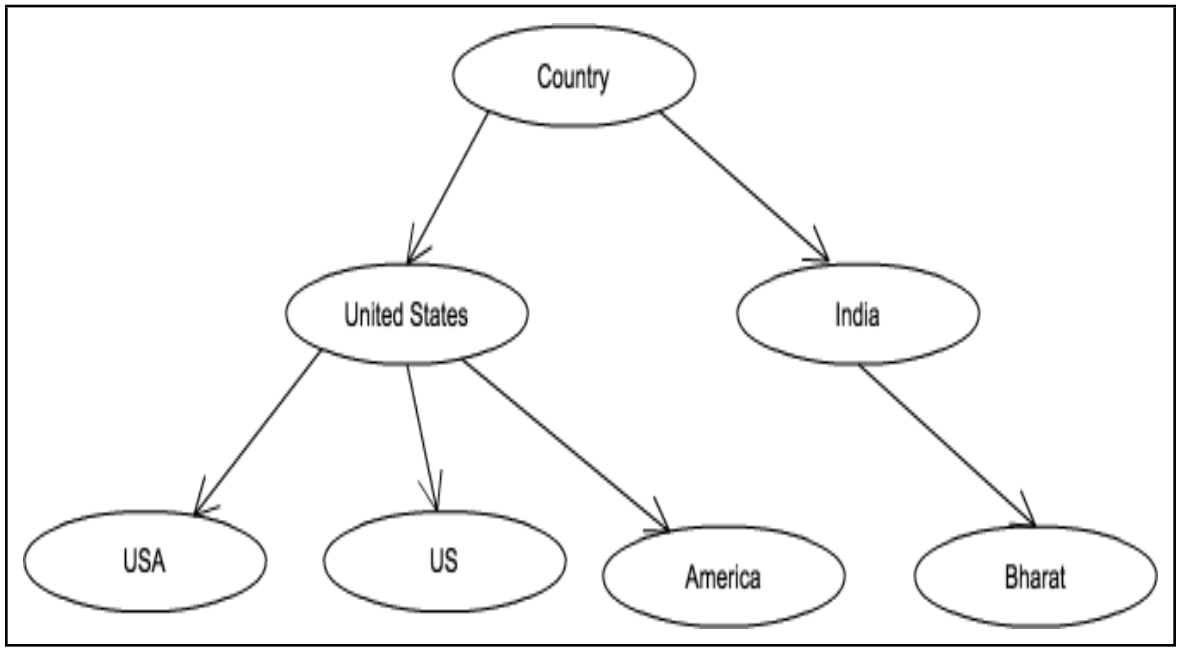


Figure 3.3: **Country Ontology**

A synonym FD $X \rightarrow_{syn} Y$ is satisfied if two tuples t_1[X] = t_2[X] then t_1[Y] $\sim$ t_2[Y] where $\sim$ is the synonym relationship in S. Intuitively, if t_1 and t_2 share the same

values in X, then their values in Y are synonyms, i.e, semantically equivalent in S.

For example, from Figure 3.3 (country synonym ontology), we understand that countries US, USA and *America* all represent the same country *United States*.

Inheritance relationship

A relation **r** satisfies an *inheritance ontology FD* $X \rightarrow_\theta Y$, if for each attribute $A \in Y$, for each $x \in \Pi_X(\mathbf{r})$, there exists a class C, such that $\Pi_A(g_x[X]) \subseteq ancestors(C)$. Let ancestors(C) be a set of all string representations for the class or any of its ancestors, i.e., $ancestors(C) = s \mid s \in canonical\ names(C)$ *or* $s \in canonical\ names(C_i)$, *where* C_i *isa* C_{i-1}, ..., C_1 *isa* C. In addition, for each $v \in \Pi_A(g_x[X])$, distance of $(v, LCA) \leq \theta$.

Similarly for inheritance relationship, we define $X \rightarrow_\theta Y$. An instance r satisfies the inheritance FD if for every equivalence class $x \in \pi_x(r)$, all tuples in $\Pi_A(g_x[X])$ are descendants of the same least common ancestor (LCA). Least Common Ancestor (LCA) of two nodes u and v in a tree is the lowest (i.e. deepest) node that has both v and w as descendants. Furthermore, to identify closely related values in the ontology, we require the distance between each value in $\Pi_A(g_x[X])$ and the LCA to be within a distance of θ.

An inheritance FD $X \rightarrow_\theta Y$ is satisfied if two tuples t_1[X] = t_2[X] then their

corresponding t_1[Y] and t_2[Y] are closely related to each other within threshold limit θ in the ontology S.

For example, from Figure 1.1 (*Medical Inheritance ontology*), the medicines *Advil* and *Paracetamol* are a type of *Acetaminophen*, which in turn are a type of *Analgesic*. As discussed earlier, θ is the permissible distance that is allowed between each node and its LCA. For example, if $\theta = 2$, then the nodes *Advil* and *Paracetamol* share an inheritance relationship as the distance between them and their LCA *Acetaminophen* is 1, which is $\leq \theta$.

When all classes in the ontology have a string representation, then synonym FDs subsume traditional FDS. i.e., $|synonyms(C)| = 1$. If $\theta = 0$, then an inheritance FD becomes a synonym FD, thus, inheritance FDs also subsume traditional FDs.

In the next section, we describe the attribute partition model and how OFDs are generated. We then present the pruning strategies that we adopt to narrow the search space and to eliminate the unlikely candidates.

Chapter 4

Discovery Algorithms

In this section, we describe how we generate candidate OFDs, and present our discovery algorithms. For clarity, we will refer to OFDs based on the synonym relationship as synonym FDs, and OFDs based on the inheritance relationship as inheritance FDs.

4.1 Generating Candidates

Synonym FD and Inheritance FD discovery algorithms traverse the search lattice from singleton sets of attributes and works its way to larger attribute sets, thereby forming the level by level search lattice. In Figure 4.4, we have a search lattice with 3 nodes, namely A, B and C. Each level in the search lattice is represented by k, where $k = 1$ to n-1, where n is the number of attributes in the relation. Every node in the attribute lattice represents an attribute set and an edge exists between X and Y if $X \subset Y$ and Y has exactly one more attribute than X, i.e., $Y = X \cup A$.

To find OFDs in r, we search through the space of non-redundant candidates, where each candidate is of the form X$\rightarrow_{syn}$Y or X$\rightarrow_{\theta}$Y. Any candidate is non-redundant if its attribute sets X,Y are not supersets of already discovered OFDs. Similar to traditional FDs, we normalize all OFDs to have single consequent in its RHS, i.e. $X \rightarrow A$. An example of attribute lattice is given in Figure 4.4 which has three attributes and three levels.

Both our algorithms follow the BFS approach but they do not necessarily traverse all the levels in the search lattice. This is due to the optimisation strategies described in Chapter 5. For example, if we discover an OFD X$\rightarrow$Y, then we prune all the edges that contain the supersets of X. Hence, these edges that are pruned will not be traversed by Synonym FD and Inheritance FD discovery algorithms. Similarly, by using other optimisations in *Chapter 5*, we avoid visiting all edges and levels in the search lattice.

All the attributes in r are modelled in the search lattice and we get n number of levels for n attributes. If we have n attributes, we will have $2^{|n|}$ nodes in the search lattice. We traverse all the edges in the lattice unless they are pruned by any of our pruning strategies.

4.1.1 Synonym FDs

Given the ontology S, two concepts c_1 and c_2 in S are synonyms if they are semantically equivalent. We develop an algorithm that discovers *Synonym FDs* that

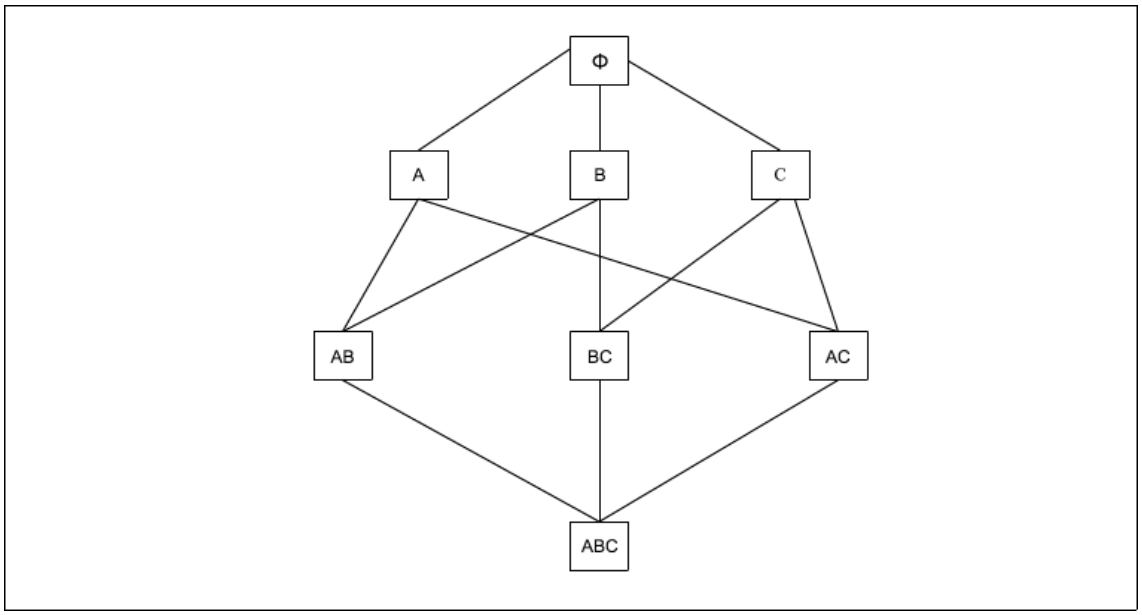


Figure 4.4: **Search lattice for three attributes A,B and C**

takes as input a relation r, and ontology S, and returns synonym FDs. Synonym FDs are of the form $X \rightarrow_{syn} Y$. As we have discussed earlier, discovered synonym FDs $X \rightarrow_{syn} Y$ is such that if two tuples $t_1[X] = t_2[X]$ then $t_1[Y] \sim t_2[Y]$ where $\sim$ is the synonym relationship in S. In *Algorithm 1*, we use canonical values for unique identification of each value in every attribute x_i in X.

Algorithm 1: Discover Synonym FDs presents the pseudocode for discovering the *synonym FDs* in relation r. *Algorithm 2: Calculate Overlap* presents the pseudocode to check whether each equivalence class in Π_X, and for all tuples within the equivalence class , we check whether each corresponding value in Y maps to the same canonical value under some common class (sense).

Discover Synonym FDs checks whether we have a synonym FD of the form $X \rightarrow_{syn}$

Y and if a synonym FD is discovered, it prunes the supersets of its LHS. In *Calculate Overlap*, let x_i represents an equivalence class in partition Π_X, where $i = \{1....|\Pi_X|\}$, where $|\Pi_X|$ is the number of equivalence classes. For each tuple $t \in \Pi_A(g_x[X])$, we check whether the corresponding canonical classes are either equal or synonyms. If this is true, then *Algorithm 1* prunes the supersets of X. Otherwise, if there exists a tuple with a canonical value not equal to other tuples within the same class, then we have found a non-equal value. The candidate OFD is then considered in the next $k+1$ level of the search lattice by expanding X with an additional attribute, $((X \cup B) \rightarrow_{syn} Y)$.

Algorithm 1 Discover Synonym FDs

INPUT r, S, π_X, canonical names (c_A), synFDs = {}

1: Initialise(U)
2: **for** each Π_X **do**
3: **for** each C_A **do**
4: OFDFound = CalculateOverlap(Π_X, C_A)
5: **if** OFDFound **then**
6: synFDs += X $\rightarrow_{syn}$ A where A $\in$ (R \X)
7: prune candidates XY $\rightarrow_{syn}$ A, $\forall$ Y $\subseteq$ (R \X)
8: **return** SynFDs

EXAMPLE 1. Consider candidate synonym FD O: Countrycode $\rightarrow$ Country

from Table 1. We have $\Pi_{Countrycode}$ = {{1,4,9}{2,6,10}{3}{5,7,8}} and $\Pi_{Country}$ = {{1}{2,6,10}{3}{4}{5,8}{7}{9}}. The first class in $\Pi_{Countrycode}$ is {1,4,9}. The country values of 1,4,9 are *UnitedStates*, *America* and USA. All these values are syntactically different. Hence we check for their canonical names of 'United States', 'America' and 'USA'. From the country ontology, we understand that they resolve to a non-empty value as three values refer to the same country 'United States'. The second equivalence class {2,6,10} represents the same country 'Canada'. The third equivalence class is {3}, since it contains a single tuple, there is no conflict. The last equivalence class is {5,7,8}. From the country ontology, their canonical names refer to the same country 'India'. Hence all the equivalence classes of $\Pi_{Countrycode}$ refers either to equal or synonymous values of *country*, the synonym FD O holds over the relation.

Algorithm 2 CalculateOverlap(Π_X, C_A)

1: $g_x[X] = \{t \mid t \in \mathbf{r} \text{ and } t[X] = x\}$
2: **for** each $x \in \Pi_X(r)$ **do**
3: **for** each $t \in \Pi_A(g_x[X])$ **do**
4: if canonical classes $\{C_{t_1} \cap \cdots \cap C_{t_m}\} = \emptyset$ then
5: **return** false
6: **return** true

4.1.2 Inheritance FDs

We develop an algorithm that takes as input a relation r, ontology S, threshold θ, and returns inheritance FDs. Inheritance FDs are of the form O: $X \rightarrow_\theta Y$. As we have discussed earlier, discovered inheritance FDs $X \rightarrow_\theta Y$ is such that if two tuples $t_1[X] = t_2[X]$ then their corresponding $t_1[Y]$ and $t_2[Y]$ share an inheritance relationship if $t_2[Y]$ is a subclass of $t_1[Y]$, or equivalently, $t_1[Y]$ is a superclass of $t_2[Y]$.

Algorithm 3 determines whether O is a qualifying inheritance FD. The candidate O holds over r if for every equivalence class, the corresponding values in Y all share a common ancestor. Furthermore, we consider a restricted version of an inheritance FD that identifies relevant relationships within a desired distance θ. Specifically, we compute the least common ancestor (LCA) between all values in the partition of Y, and check whether the distance between each of the values and the LCA is within θ.

Algorithm 3 Discover Inheritance FDs

INPUT r, S, π_X, Threshold θ, inheritanceFDs = {}

1: **for** each Π_X **do**

3: **for** each C_A **do**

4: IsFound = ComputeLCA(Π_X, C_A)

5: **if** OFDFound **then**

6: isaFDs += X $\rightarrow_\theta$ A where A $\in$ (R \X)

7: prune candidates XY $\rightarrow_\theta$ A, $\forall$ Y $\subseteq$ (R \X)

8: **return** isaFDs

Algorithm 3: Discover Inheritance FDs checks whether we have an inheritance FD of the form $X \rightarrow_\theta Y$ and if an an inheritance FD is discovered, it prunes the supersets of its LHS. In *Algorithm 4: ComputeLCA*, for each equivalence class in Π_X, we identify the corresponding values in Π_Y. For each of these equivalence classes in Π_Y, we compute the least common ancestor among all its tuples t. If the distance between each t and the LCA is within the threshold θ, then the candidate X $\rightarrow_\theta$ Y holds. If this is true, then *Algorithm 3* prunes the supersets of X. Otherwise, if there exists a tuple with a canonical value not equal to other tuples within the same class, then we have found a non-equal or non-synonymous value. The candidate OFD is then considered in the next $k+1$ level of the search lattice by expanding X with an additional attribute, $((X \cup B) \rightarrow_\theta Y)$.

Algorithm 4 ComputeLCA(Π_X, C_A)

1: $g_x[X] = \{t \mid t \in \mathbf{r} \text{ and } t[X] = x\}$

2: **for** each $x \in \Pi_X(r)$ **do**

3: **for** each $t \in \Pi_A(g_x[X])$ **do**

4: if $LCA\ (ancestor(C_{t_1}) \cdots ancestor(C_{t_n})) \geq \theta$ then

5: return false

6: **return** true

EXAMPLE 2. Consider candidate inheritance FD [Symptoms, Diagnosis] $\rightarrow_\theta$ [Medicine], and the ontology in Figure 1.1. For each $x \in \Pi_X$ = {{1,2,3}{4,5,6}{7,8,9,10}}, we identify the corresponding values in $\Pi_A(g_x[X])$. For x1 = {1,2,3} and x2 = {4,5,6} the LCA is analgesic. Suppose θ = 2, all values in $\Pi_A(g_x[X])$ are within a distance of θ, e.g., distance(Pepcid, Analgesic') = distance(Advil, Analgesic) = distance('Paracetamol','Analgesic') = 2. Hence, for θ = 2, the inheritance FD holds.

4.1.3 Complexity Analysis

The worst case complexity of both the algorithms with respect to the number of attributes is exponential as there are $2^{|n|}$ nodes. However with respect to number of tuples, worst case time complexity of Synonym FD algorithm is quadratic. Worst case time complexity of Inheritance FD algorithm is cubic as it scans different levels of ontology based on varying θ. The complexity increases with the increased number of levels in the ontology tree. Hence, on an average case, time complexity of both

Synonym and Inheritance FDs is *O(log k/2)*, where k is the number of levels in the search lattice.

Since the solution space for minimal ontology FDs is exponential, a polynomial time algorithm in the number of attributes cannot exist. The same conclusions have been reached for the discovery of traditional FDs and inclusion dependencies. However, the recent algorithm for the discovery of order dependencies (OD) [8] has a factorial worst-case time complexity. This is because ODs are defined over lists of attributes. Hence the search space is represented as a lattice of attribute permutations with $\lfloor n! * e \rfloor$ nodes, where e is the exponential function.

Chapter 5

Optimisations

In this chapter, we present our optimisation strategies that improve the efficiency of our Synonym FD and Inheritance FD discovery algorithms.

In our work, we have adapted these optimisation strategies that improve the performance and efficiency of our algorithms. By following these optimisation strategies, we avoid pruning the redundant candidates by avoiding visit to all the edges in the search lattice. We cannot guarantee that we prune all the time, hence our algorithms may sometimes visit all edges in the search lattice.

Lemma 5.1

Rule of augmentation:

Let r be a relation consisting of attributes X, Y and indepdendent attribute A. If $X \rightarrow A$ is satisfied over r, then $XY \rightarrow A$ is satisfied for all $Y \subseteq r \setminus X$.

Rule of augmentation states that if we identify an OFD $X \rightarrow A$, then we prune all the supersets of LHS of the OFD. Thereby, we avoid visiting all edges and levels in the search lattice. This optimisation strategy has improved the performance of our algorithms by an average of 3x times on both clinical trials and census datasets.

EXAMPLE 3. If we have an OFD $[countrycode] \rightarrow_{syn} [country]$, then we prune all the supersets of *countrycode*. Some of the supersets include $[countrycode, Symptoms]$ '$tarrow_{syn}$ $[country]$, $[countrycode, Diagnosis] \rightarrow_{syn} [country]$, $[countrycode, Symptoms, Diagnosis] \rightarrow_{syn} [country]$ and so on.

Lemma 5.2

Key pruning: Let r be a relation consisting of set of attributes X,Y and independent attribute A. If X is a key in r, then for any attribute A, $X \rightarrow A$ is satisfied in r.

If we find a key attribute X in a relation r, then all tuples of X map to unique values with all other attributes in the relation. Also, since X is a key in r, the size of each partition in π_X is equal to one. Furthermore, when X is a key in r, then it is a Functional Dependency, which in turn is an Ontology Functional Dependency.

EXAMPLE 4. From *Table* 1.1, we understand that $\Pi_{Medicine} = \{\{1\}\{2\}\{3\}\{4\}\{5\}\{6\}\{7\}\{8\}\{9\}\{10\}\}$. Here all equivalence class of $\Pi_{Medicine}$ is of size 1, hence all values of $Medicine$ map to a unique value in its RHS.

Lemma 5.3

If $A \subset Z$, then $Z \rightarrow A$ is a trivial dependency.

Lemma 5.4

Rule of FD Discovery: If $|\Pi_A(g_x[X])| = 1$ for an $x \in \pi_X$, then all values in x map to a unique value in $\Pi_A(g_x[X])$.

If all the values in LHS of the OFD correspond to the same value in RHS, then it is not necessary to check for a synonym or inheritance relationship, as a traditional FD is satisfied. All the traditional FDs are considered to be OFDs because entities are said to be equivalent if they are identical.

EXAMPLE 5. Let us consider that relation r contains attributes A and B. Equivalence class Π_B in $B \rightarrow A$ is {{1}{3}{2,4}{6}{5,7,8}{9}{10}}. If all the tuples in Π_B map to a single value in A, then a traditional FD is satisfied as all tuples in LHS map to a single value in RHS.

Chapter 6

Experimental Evaluation

We tested the synonym FD and inheritance FD algorithms on two real datasets (clinical trials [17] of 1M records and census data [19] of 150K records). Both the experiments show that our algorithms scale well for large datasets. We have conducted scalability experiments with respect to varied number of tuples and attributes in the relation. In addition, we have conducted performance evaluation of our optimisation techniques and comparative evaluation with a well-known lattice-based FD algorithm, TANE.

6.1 Datasets

Setup: Our implementation was written using Java 1.7. All accuracy and comparison experiments were run on a server with four virtual CPUs (2.1 GHz each) and 32 GB of memory. The performance experiments were run on a server with eight virtual CPUs (2.1 GHz each) and 32 GB of memory.

Real Datasets: Clinical trials dataset from LinkedCT.org provides an open interface for international clinical trials data. The XML version of this data was transformed into open linked data [17]. Clinical trials dataset has fifteen attributes and 1M tuples. We used a portion of this dataset that includes data from six countries namely United States, Canada, Germany, Japan, India and Australia. Ontologies for clinical trials dataset include medicine ontology [15] and country ontology [16] as shown in Figures 1.1 and 3.3.

The census dataset has 150K tuples and eleven attributes [19]. The attributes of census dataset include work class, native country, marital status, race, relationship, capital gain, capital loss, work hours per week, salary scale, occupation and loans. Census dataset contains the census records of six different countries namely Canada, United States, Germany, Cuba, Mexico and Jamaica. Ontologies for census dataset include country ontology in Figure 3.3 and occupational ontology [20] in Figure 6.5.

6.2 Scalability Experiments

We used the clinical trials and census datasets to test the performance of Synonym FD and Inheritance FD algorithms. We evaluated the scalability of our algorithms to assess the running time performance as we varied the number of tuples (N) and the number of columns (n).

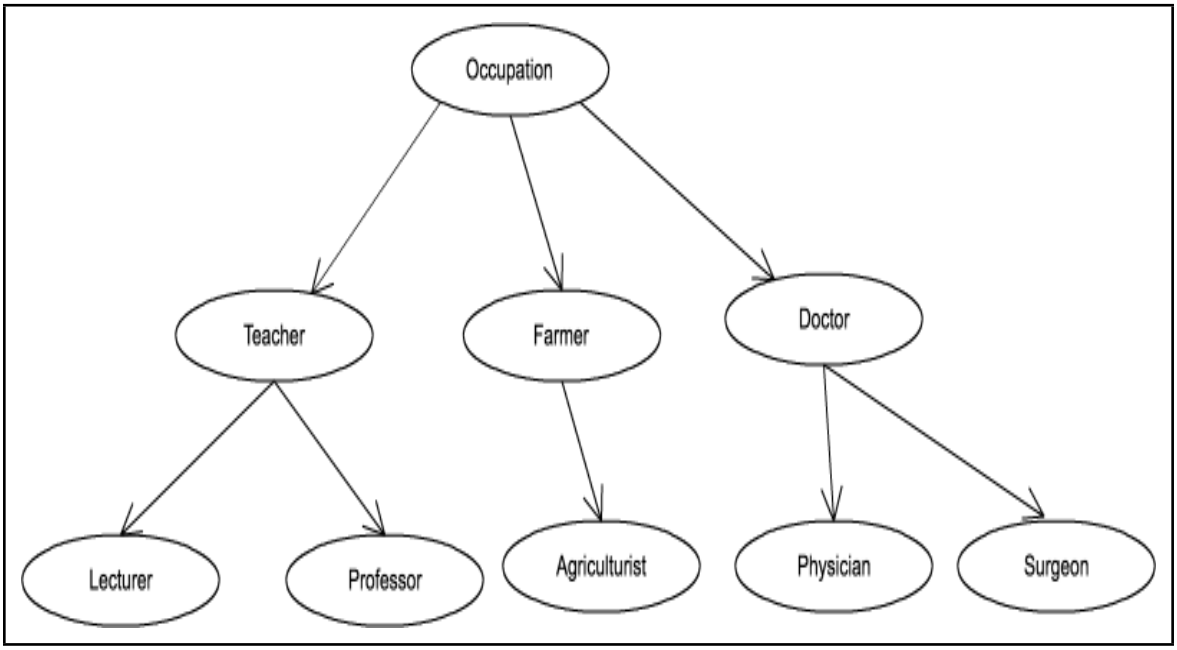


Figure 6.5: **Occupational Ontology**

6.2.1 Scalability in N

We vary the number of tuples from 150K to 1M and measure the running time of both Synonym FD and Inheritance FD algorithms. We run the algorithms six times and measure their average as the reported running time. We have compared the performance of our algorithms with TANE, a well-known lattice based FD discovery algorithm. The execution times calculated were the average over six runs, not including input/output operations.

Figure 6.6 shows the results for the scalability experiment on the Clinical trials dataset with 1M rows for Synonym FD, Inheritance FD and TANE algorithms. Medical ontology (Figure 1.1) has five levels in the hierarchy. We have tested the threshold or edge length (θ) for $\theta \leq 5$. While discovering Synonym FDs, we traverse

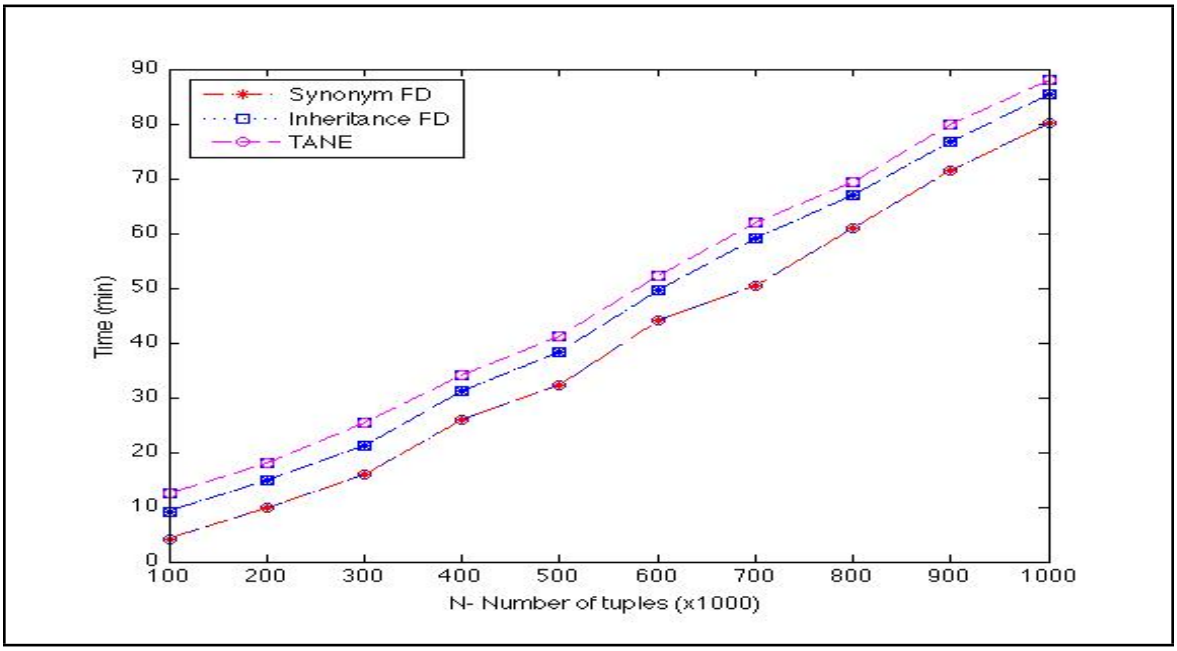


Figure 6.6: **Scalability in N - Clinical trials dataset**

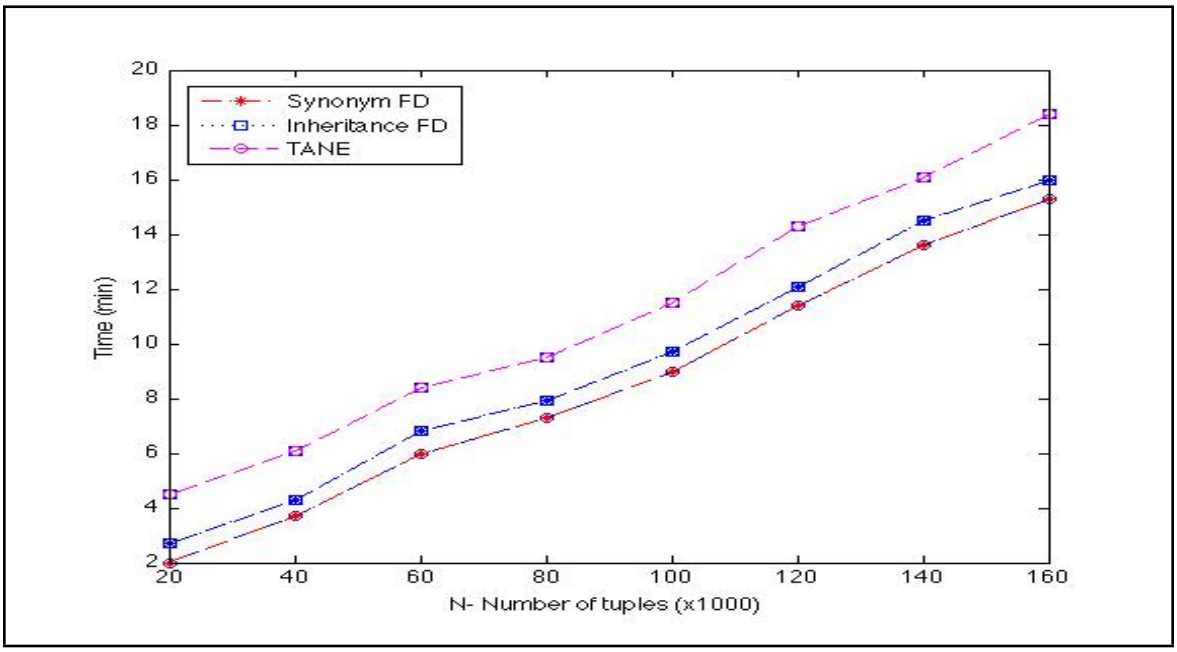


Figure 6.7: **Scalability in N - Census dataset**

only the leaves of the Ontology Tree, whereas for Inheritance FDs, we traverse the levels of the Ontology Tree based on the value of θ. We can infer from Figure 6.6 that Inheritance FDs take approximately five to six minutes more than that Synonym FDs. The discovery of increased number of Inheritance FDs compared to Synonym FDs is due to the traversal of increased number of levels in the Ontology Tree based on the value of θ.

Figure 6.7 shows the results for the scalability experiment on the census dataset with 150K rows for both Synonym FD and Inheritance FD algorithms. Here, we have used the threshold limit θ to be ≤ 3. Both Country and Occupational ontology have three levels in the hierarchy. We can infer from Figure 6.7 that Inheritance FDs take approximately three to five minutes more than that Synonym FDs.

In addition, our algorithms outperform the TANE by an average of 19.5%. This is because our optimization strategies in Section 5 achieve an average of 8.6% improvement in running time. *Lemma 5.1* in Section 5 states that, if $Z \rightarrow A$ is satisfied over r, then $ZY \rightarrow A$ is satisfied for all $Y \subseteq R \setminus Z$. For instance, if we detect an Ontology FD $AB \rightarrow C$, then we do not have to verify the Ontology FD $ABD \rightarrow C$ as all OFDs with supersets of AB also hold in r . This is not the case with traditional FDs, because for traditional FDs TANE has to verify if $ABD \rightarrow C$ holds, $ABDE \rightarrow C$ holds and so on. This incurs extra cost for traditional FDs.

From *Figure 6.6* and *Figure 6.7*, we understand that both the graphs show linear performance, as opposed to the polynomial performance as discussed in the *Section*

4.1.3. This is because we have run the algorithms six times and took their average as the reported running time. Averaging over six runs has resulted in the linear performance of algorithms.

In a nutshell, our experiments show that both Synonym FD and Inheritance FD algorithms scale well with several thousands of rows in both the datasets.

6.2.2 Scalability in n

In this section, we show how Synonym FDs and Inheritance FDs behave on tables with an increasingly large number of columns.

We considered N to be 100K and $\theta \leq 5$. Figure 6.8 shows the column scalability experiment on the Clinical trials dataset for different n. We have discovered eighteen OFDs for synonyms relationship and twenty four OFDs for inheritance relationship. The execution time more than triples when we move from n=7 to n=15. This is due to the inherent complexity of OFD detection as we increase the n from 7 to 15. The increased number of attributes cause a large number of edges to be evaluated in the search lattice while discovering the Synonym and Inheritance FDs.

The column scalability experiment on census dataset is depicted in Figure 6.9. We have discovered nine OFDs for synonyms relationship and fifteen OFDs for inheritance relationship. Here the calculation of partitions are comparatively faster than Clinical trials dataset because of the lesser number of attributes.

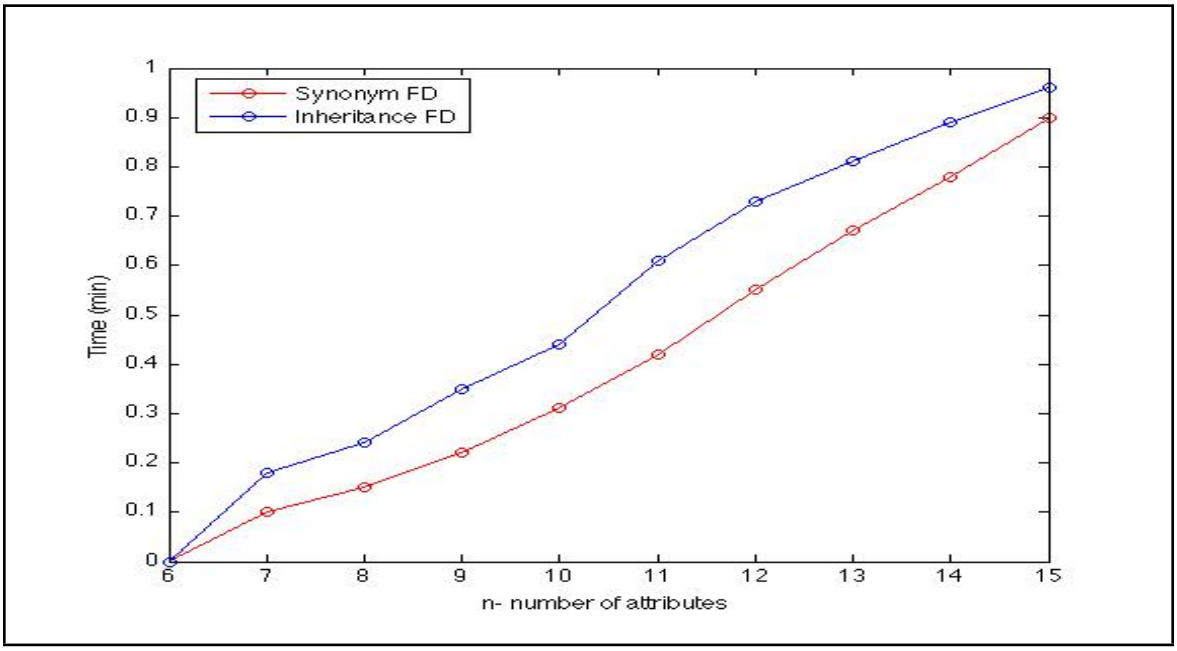


Figure 6.8: **Scalability in n - Clinical trials dataset**

From Figures 6.8 and 6.9, the execution time increases as we increase the number of columns along the x axis. We also observed that for the small number of attributes ($n < 10$), execution time is less compared to the large number of attributes ($n > 10$) because the number of combinations to be computed in the search lattice are comparatively less.

6.3 Optimisation Strategies

In this section, we evaluate the performance of our optimisation strategies on the clinical trials and census datasets.

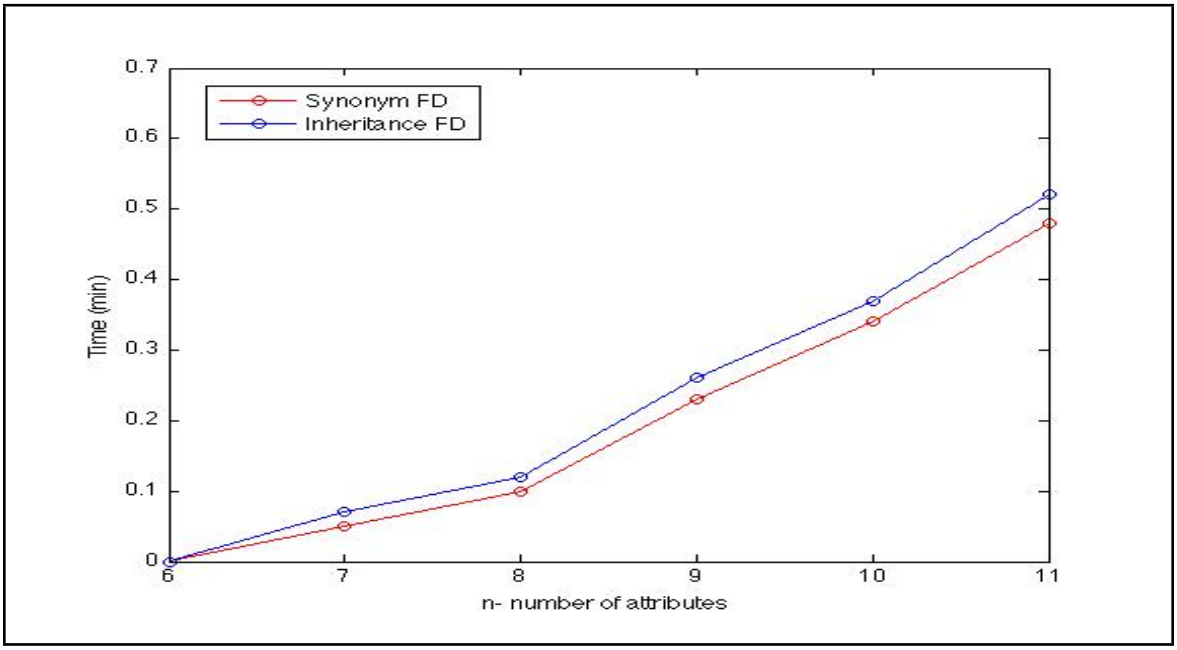


Figure 6.9: **Scalability in n - Census dataset**

Lemma 5.1 states that if $Z \rightarrow A$ holds over r, then all OFDs with supersets of Z also hold in r. Hence, we prune the supersets of Z. Figure 6.10 shows the performance of Synonym FD and Inheritance FD algorithms for clinical trials dataset based on Lemma 5.1. From Figure 6.10, we observe that Lemma 5.1 has increased the performance of our algorithms by an average of 3x times.

Lemma 5.2 states that if Z is a key in r, then for any attribute A, $Z \rightarrow A$ is satisfied in r. In our clinical trials and census datasets, we did not encounter an attribute that is a key for that relation. Hence, we did not evaluate the Lemma based on key pruning.

Lemma 5.4 states that if all values of an equivalence class x maps to the same value A in an OFD, then it is not necessary to check for Synonym or Inheritance

relationship as a traditional FD is satisfied in x. From Figure 6.11, we observe that Lemma 5.4 has improved the performance of our algorithms by an average of 2x times.

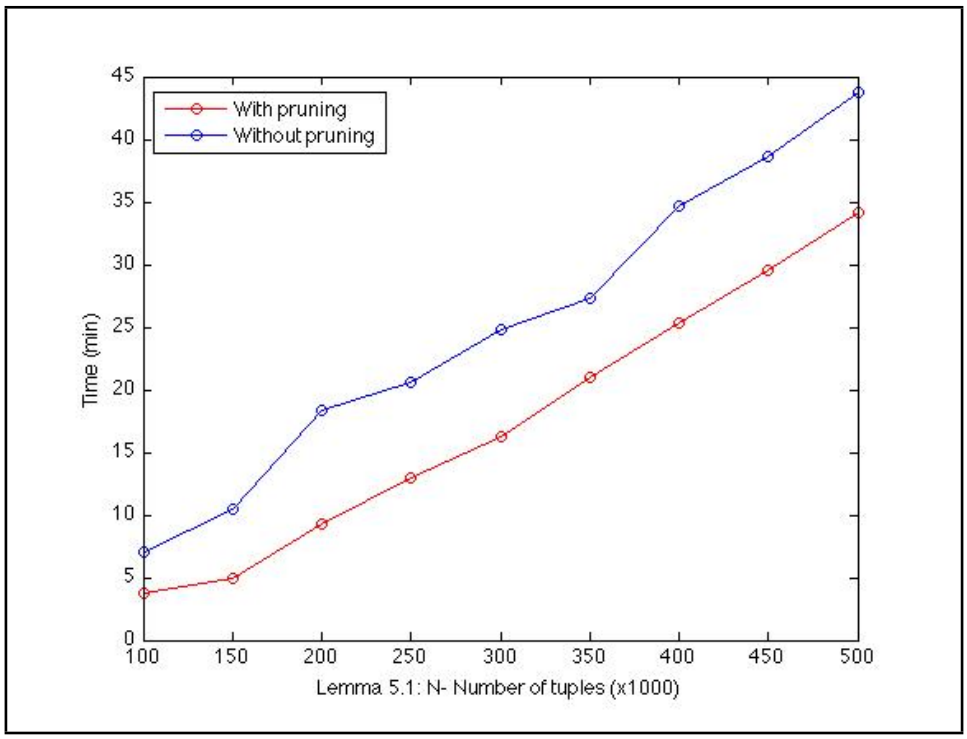
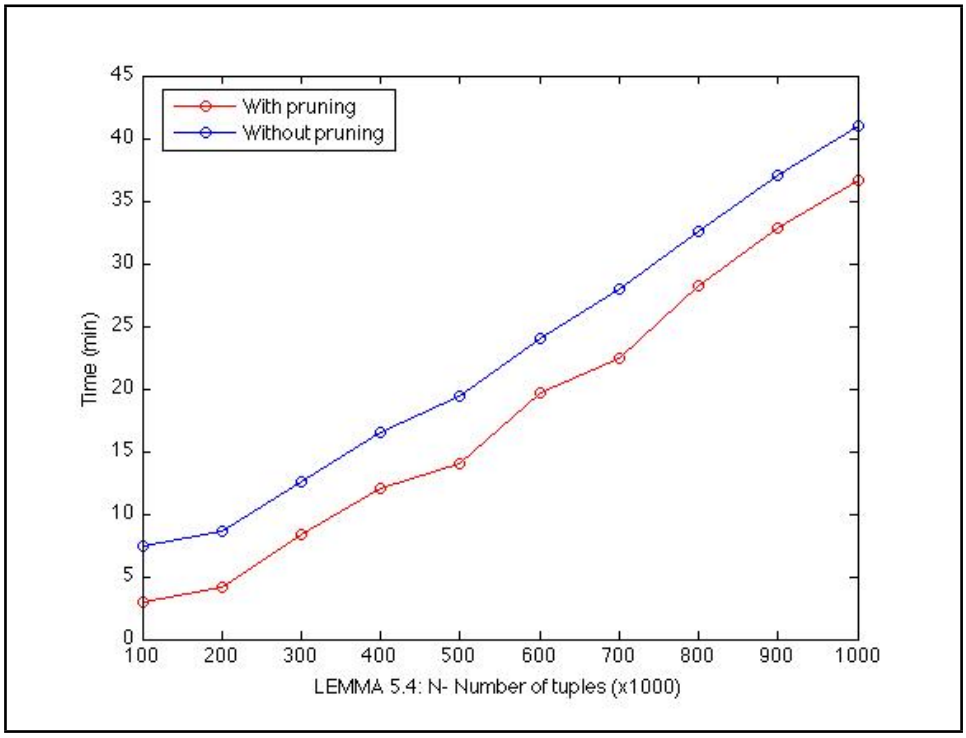


Figure 6.10: **Performance evaluation based on rule of augmentation - Clinical trials**

6.4 Qualitative Evaluation

We evaluate the quality of the discovered OFD rules using real datasets. Specifically, we measure their quality using *precision* and *recall* .

Precision is defined as the ratio of number of correct OFDs to the number of returned OFDs. Recall is defined as the ratio of number of correct OFDs to the number of



]

Figure 6.11: **Performance evaluation based on rule of FD discovery - Clinical trials**

true OFDs. To measure the accuracy and completeness of the discovered OFDs, we used clean versions of the data containing a known number of satisfying Synonym and Inheritance FDs as the ground reality. We then randomly injected 2% error in the datasets and measured the precision and recall of the discovered OFDs. We found all the discovered dependencies to be correct (leading to 100% precision), and the recall values ranged between 70% to 80%, as shown in Table 6.4.

The measures of precision and recall concentrate the evaluation on the return of true positives, asking what percentage of the relevant documents have been found and how many false positives have also been returned. We get a precision of 100% as the number of OFDs returned by our algorithms for dirty dataset is a subset of number of OFDs returned by our algorithms for the clean version of our dataset. To reiterate, we have injected 2% error in the datasets. So, technically we expect the recall to be exactly 80%. But in our case, recall varies between 70% to 80% because the OFDs returned by the dirty dataset is approximately 30% more the clean dataset.

$$\text{Precision} = \frac{\#correct}{\#returned}$$

$$\text{Recall} = \frac{\#correct}{\#true}$$

Dataset	Algorithm	Precision	Recall
Clinical trials	Synonym FD	100%	76%
Clinical trials	Inheritance FD	100%	73%
Census	Synonym FD	100%	73%
Census	Inheritance FD	100%	80%

Table 6.4: Precision and Recall

6.5 Comparative Evaluation

There are no equivalent techniques for discovering OFDs. However, there are algorithms for discovering Functional Dependencies (FDs). TANE is a lattice-based FD discovery algorithm. In relation to our work, TANE discovers FDs that hold on any relation r. We compare our techniques against TANE to evaluate the number of FDs returned by TANE against number of OFDs returned by Synonym FD and Inheritance FD algorithms.

Algorithm	Clinical trials	Census
Synonym FD	16	7
Inheritance FD	21	10
TANE (FD)	4	3

Table 6.5: Comparative Evaluation

Table 6.5 shows the number of dependencies returned by our discovery algorithms and TANE. We have considered the $N = 100$K. From Table 6.5, we observe that our discovery algorithms identify approximately 3x more OFDs than FDs. In synonym FD discovery, there exists multiple representations of the same entity that are not all syntactically equivalent. For example, USA, States and America are different representations of the same country, United States. These multiple representations lead to multiple (and an increased number) of Synonym FDs being discovered. In Inheritance FD discovery, we check whether all values in the RHS of the FD share an inheritance relationship in the ontology. For example, the medicines Pepcid and Zantac are Histamine, which in turn is an Analgesic. This is in contrast to FDs which would identify multiple associations to the same entity as erroneous.

Even though our algorithms discover close to 3x more dependencies than TANE, our algorithms achieve faster running time when compared to TANE. It is because of our efficient optimisation strategies which improve the performance of our algorithms by an average of 2x to 3x times. In both Clinical Trials and Census datasets, our optimizations are more effective. *Lemma 5.1* {*Rule of augmentation*} is the most effective optimisation strategy because it has improved the performance of our algorithms by an average of 3x times.

Chapter 7

Conclusion and Future Work

Current data cleaning algorithms do not discover constraints based on semantic equivalence. We have taken a step forward to find dependencies in a relational database based on semantics of domain specific knowledge in ontologies. We have proposed a new class of dependencies, Ontology Functional Dependencies that can be used to capture domain specific relationships for data cleaning. Domain specific relationships are stored in ontologies. Our dependency discovery algorithms identify attribute values satisfying Synonym and Inheritance relationships in a relational instance. Our optimisation strategies have improved the performance of our algorithms by an average of 2x to 3x times. In addition, our experiments reveal that our algorithms can be applied to provide a set of data quality tools that enforce a broader set of attribute relationships.

7.1 Future Work

There is naturally much to be done. Currently, we only consider complete OFDs that hold over the entire dataset. In future, we can consider the notion of Conditional OFDs, that hold over a portion of the relation for certain specific values. In our current work, we discover an OFD $A \rightarrow B$ only if all the partitions of π_A map to an unique canonical value in B. In discovery of Conditional OFDs, we can check whether some equivalence classes in π_A map to an unique canonical value. Let x_i represent an equivalence class in partition Π_A, where $i = \{1....|\Pi_A|\}$, where $|\Pi_A|$ is the number of equivalence classes. For each tuple $t \in \Pi_A(g_x[X])$, we check whether the corresponding canonical classes are either equal or synonyms. In Conditional OFDs, this condition need not be true for all partitions in Π_A. Even when some partitions in Π_A map to an unique canonical class, then a Conditional OFD is discovered.

We also intend to consider extensions to other relationships such as type-of, and the use of ontologies to discover other types of data quality rules such as conditional FDs and denial constraints. In our current work, we compare the performance of our algorithms with TANE, which is a lattice based FD Discovery algorithm. In future, we can compare our discovery algorithms with various other FD discovery algorithms (e.g., FD Mine) and compare them in terms of data repair quality and run-time performance. Furthermore, we can also apply the supervised machine learning algorithms to learn the semantic patterns from the given concepts in the ontology tree and use it to predict the semantics of other related concepts while discovering OFDs in the relation.

Bibliography

[1] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. *A cost-based model and effective heuristic for repairing constraints by value modification.*. In *SIGMOD*, pages 143−154, 2005.

[2] X. Chu, I. F. Ilyas, and P. Papotti. *Holistic data cleaning: Putting violations into context.*. In *ICDE*, pages 458−469, 2013.

[3] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. *Improving data quality:Consistency and accuracy.*. In *VLDB*, pages 315−326, 2007.

[4] Y. Huhtala, J. Kinen, P. Porkka, and H. Toivonen. *Efficient discovery of functional and approximate dependencies using partitions.*. In *ICDE'98*, pages 392−401, 1998.

[5] S. Judah and T. Friedman. *Twelve ways to improve your data quality. Gartner Research Report*, 2014.

[6] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. *Metric functional dependencies.* In *ICDE*, pages 1275−1278, 2009.

[7] P. Langer and F. Naumann. *Efficient order dependency detection.* In *VLDB*, pages 223−241, 2016.

[8] N. Prokoshyna, J. Szlichta, F. Chiang, R. Miller, and D. Srivastava. *Combining quantitative and logical data cleaning.* In *PVLDB*,pages 300−311, 2015.

[9] F. Chiang, R.J,Miller. *Discovering data quality rules.* In *VLDB* pages 1166−1177, August 2008.

[10] Bruggemann. *Rule mining for automatic ontology based data cleaning* In *Proceedings of 10th Asia-Pacific Web Conference*, April 2008.

[11] S. Song and L. Chen. *Differential dependencies: Reasoning and discovery.* In *ICDE'11*, pages 131−162, 2011.

[12] L. Golab, H. Karloff, F.Korn, A. Saha, and D. Srivastava. *Sequential dependencies.* In *PVLDB*, pages 574−585, 2009.

[13] Alexander Maedche and Steffen Staab. *Ontology Learning for the Semantic Web. Research Report in FZI Research Center for Information Technologies, University of Karlsruhe, Germany.*

[14] Mena, E., Kashyap, V., Sheth, A. and Illarramendi. *A. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies.* In *PVLDB*, pages 574−585, 2010.

[15] http://medicine-ontology.org/ *Institute for Genome Sciences*

[16] https://countrycode.org/. *Country Codes, Phone Codes, Dialing Codes, Telephone Codes and ISO Country Codes list*

[17] http://www.nimh.nih.gov/funding/clinical-research/datasets/nimh-procedures-for-requesting-data-sets.shtml *Clinical Trials dataset*

[18] https://mor.nlm.nih.gov/ *Medical Ontology Research*

[19] United States Census Bureau. *http://census.gov/*

[20] https://wordnet.princeton.edu/. *Word Net*

[21] https://archive.ics.uci.edu/ml/datasets.html *UCI Machine learning repository*

[22] L.V.S.Lakshmanan, R.T.Ng, J.Han, and A.Pang. *Optimization of constrained frequent set queries with 2-variable constraints* In *SIGMOD'99*, pages 157−168.

[23] S. Lopes, J.-M. Petit, and L. Lakhal. *Efficient discovery of functional dependencies and armstrong relations.* In *EDBT'00*, pages 350−364, 2000

[24] C. Wyss, C. Giannella, and E. L. Robertson.. *Fastfds: A heuristic-driven, depth-first alg for mining fds from relations.* In *DaWaK'01*, pages 101−110, 2001.

[25] N. Novelli and R. Cicchetti. *FUN: An efficient algorithm for mining functional and embedded dependencies. ICDT*, pages 189−203, 2001.

[26] F. Chiang, R.J,Miller. *A Unified Model for Data and Constraint Repair*. In *Proceedings of the ICDE*, pages 446−457, April 2011.

[27] Maksims Volkovs, Fei Chiang, Jaroslaw Szlichta, Rene J. Miller *Continuous data cleaning*. In *Proceedings of the ICDE*, 2014, pages 244−255, April 2014.

[28] L. Bravo, W. Fan, and S. Ma. *Extending dependencies with conditions.* In *VLDB'07*, pages 243−254, 2007.

[29] S. Kolahi and L. V. Lakshmanan, *On approximating optimum repairs for functional dependency violations.* In *ICDT'09*, pages 301−338, 2009.