Efficient Discovery of Ontology Functional Dependencies

Sridevi Baskaran*, Alexander Keller#, Fei Chiang*, Jaroslaw Szlichta#
*McMaster University, Department of Computing and Software, Canada
#University of Ontario Institute of Technology, Faculty of Science, CS, Canada
baskas@mcmaster.ca, alexander.keller@uoit.net, fchiang@mcmaster.ca, jaroslaw.szlichta@uoit.ca

ABSTRACT

Poor data quality has become a pervasive issue due to the increasing complexity and size of modern datasets. Constraint based data cleaning techniques rely on integrity constraints as a benchmark to identify and correct errors. Data values that do not satisfy the given set of constraints are flagged as dirty, and updates are made to re-align the data and the constraints. However, many errors require user input to resolve due to domain expertise defining specific terminology and relationships. For example, in pharmaceuticals, 'Advil' *is-a* brand name for 'ibuprofen' that can be captured in a pharmaceutical ontology. While functional dependencies have traditionally been used in existing data cleaning solutions to model syntactic equivalence, they are not able to model broader relationships (e.g., is-a) defined by an ontology. In this paper, we take a first step towards extending the set of data quality constraints used in data cleaning by defining and discovering *Ontology Functional Dependencies* (OFDs). We lay out theoretical foundations for OFDs, including a set of sound and complete axioms, and a linear inference procedure. We then develop effective algorithms that discover a complete, minimal set of OFDs, and a set of optimizations that efficiently prune the search space. Our experimental evaluation using real data show the scalability and accuracy of our algorithms.

1. INTRODUCTION

Organizations are finding it increasingly difficult to reap value from their data due to poor data quality. The increasing size and complexity of modern datasets exacerbate the fact that real data is dirty, containing inconsistent, duplicated, and missing values. A Gartner Research study reports that by 2017, 33% of the largest global companies will experience a data quality crisis due to their inability to trust and govern their enterprise information [7]. In constraint based data cleaning, data dependencies are used to specify data quality requirements. Data that is inconsistent with respect to the dependencies are identified as erroneous, and updates to the data are generated to re-align the data with the dependencies. Deciding which updates to apply often goes beyond simply identifying similar or equivalent string values, as user input is necessary to gather information such as terminology, concepts, and relationships that are relevant for a given application domain.

Existing data cleaning approaches have focused on using functional dependencies (FDs) to define the attribute relationships that the data must satisfy [3, 10]. Extensions include the use of inclusion dependencies [3], conditional functional dependencies [5], and denial constraints [4]. While FDs have been the most commonly used dependencies, they are limited to identifying attribute relationships based only on syntactic equivalence.

For example, Table 1 shows a sample of clinical trial records containing patient country codes (CC), country (CTRY), symptoms (SYMP), diagnosis (DIAG), and the prescribed medication (MED). Consider two FDs: F_1: [CC] $\rightarrow$ [CTRY] and F_2: [SYMP, DIAG] $\rightarrow$ [MED]. The tuples ($t_1, t_5,\ t_6$) do not satisfy F_1 as 'United States', 'America', and 'USA' are not syntactically (string) equivalent (the same is true for ($t_2, t_4,\ t_7$)). However, we know that the 'United States' is synonymous with 'America' and 'USA', and (t_1, t_5, t_6) all refer to the same country. Similarly, 'Bharat' in t_4 is also synonymous with 'India' as it is the country's original Sanskrit name. For F_2, (t_1, t_2, t_3) and (t_4, t_5, t_6) do not satisfy the dependency as the consequent values all refer to different medications. However, upon closer inspection, with domain knowledge from a medical ontology (Figure 1), we see that the values participate in an inheritance relationship. Both 'ibuprofen' and 'naproxen' are nonsteroidal anti-inflammatory drugs (NSAID), and 'tylenol' is an 'acetaminophen' drug, which in turn is an 'analgesic'.

The above example demonstrates that real data often contains domain specific relationships that go beyond simple syntactic equivalence. It also highlights two common relationships that occur frequently between two values u and v: (1) u and v are *synonyms*; and (2) u is-a v denoting *inheritance*. These relationships are often defined within domain specific ontologies that can be leveraged during the data cleaning process to identify and enforce domain specific data quality rules. Unfortunately, traditional FDs are unable to capture these relationships, and existing data cleaning approaches flag tuples containing synonymous and inheritance values as errors. This leads to an increased number of "errors", and a larger search space of data repairs to consider.

In this paper, we take a first step to address this problem by defining a new class of dependencies called *Ontology Functional Dependencies* (OFDs) that capture relationships defined in an ontology. We focus on the synonym and is-a (inheritance) relationships between two attribute values. We make the following contributions:
(1) OFDs. We define a novel class of dependencies called OFDs based on the synonym and inheritance relationships. We present two discovery algorithms that identify a complete, minimal set of OFDs over a relation. Our automated discovery algorithms alleviate the burden of specifying these dependencies for data cleaning.

id	CC	CTRY	SYMP	DIAG	MED
t_1	US	United States	joint pain	osteoarthritis	ibuprofen
t_2	IN	India	joint pain	osteoarthritis	NSAID
t_3	CA	Canada	joint pain	osteoarthritis	naproxen
t_4	IN	Bharat	nausea	migrane	analgesic
t_5	US	America	nausea	migrane	tylenol
t_6	US	USA	nausea	migrane	acetaminophen
t_7	IN	India	chest pain	hypertension	morphine

Table 1: Sample clinical trials data.

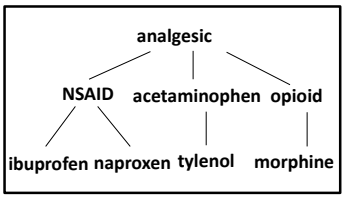

Figure 1: Ontology.

id	X	Y	Classes for Y
t_1	u	v	{C,D}
t_2	u	w	{D,F}
t_3	u	z	{C,F,G}

Table 2: Defining OFDs.

(2) Axioms and inference procedure. We introduce a set of axioms for OFDs, and prove these are sound and complete. We use our inference rules to propose optimizations that enable our discovery algorithms to avoid redundant computations. While the inference complexity of other FD extensions is co-NP complete, we show that the inference problem for OFDs remains linear. Our inference procedure can be used to reason about the consistency and correctness of data design. **(3) Optimizations.** We present a set of optimizations to prune the search space and improve the algorithm running time, without sacrificing correctness. **(4) Evaluation.** We evaluate the performance and effectiveness of our techniques using two real datasets containing up to 1M records. Our experiments demonstrate that our algorithms scale well, and outperform an existing FD discovery algorithm.

2. BACKGROUND

Definitions. A functional dependency (FD) F over a relation R is represented as $X \rightarrow Y$, where X, Y are attributes in R. An instance I of R satisfies F if for every pair of tuples $t_1, t_2 \in I$, if $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$. In the search for OFDs, we define a partition of X, Π_X, as the set of equivalence classes containing tuples with values equal to the values in X. 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 equivalence class. For example, in Table 1, $\Pi_{CC} = \{\{t_1, t_5, t_6\}\{t_2, t_4, t_7\}\{t_3\}\}$.

Given a relational instance I, our objective is to discover a minimal (non-redundant) set of OFDs. An OFD $X \rightarrow A$ is *trivial* if $A \in X$. If $X \rightarrow A$ is minimal and holds over I, then there is no dependency $Z \rightarrow A$ that holds such that $Z \subset X$. Given an ontology S, concepts and terms may be applicable only for a specific domain or limited representation within S. As such, the meaning of concepts and terms for a given S can be modelled according to different *senses* that correspond to explicit ontological distinctions. We define classes E to contain the applicable senses for a given attribute value. In addition, let *synonyms* (E) be the set of all canonical names for a given class E. Let *canonical names*(C) be the set of all classes for a given term C. Let $descendants(E)$ be a set of all string representations for the class E or any of its descendants, i.e., $descendants(E)$ = $\{s \mid s \in$ *synonyms*(E) or $s \in$ *synonyms*(E_i), where E_i *is-a* E_{i-1}, ..., E_1 *is-a* $E\}$.

We define OFDs with respect to a relationship from a given ontology S. We consider two relationships: synonyms and inheritance. For synonyms, we define an OFD O_{syn}: $X \rightarrow_{syn} Y$. An instance I satisfies O_{syn} if for every equivalence class $x \in \Pi_X(I)$, there exists a common class (sense) for all the tuples in $\Pi_A(g_x(X))$, where $g_x[X] = \{t | t \in I$ and $t[X] = x\}$, and A $\in Y$. We formalize this in the following definition.

DEFINITION 1. *A relation I satisfies a synonym FD* $X \rightarrow_{syn} Y$*, if for each attribute* $A \in Y$*, for each* $x \in \Pi_X(I)$*, there exists a class E, such that* $\Pi_A(g_x[X]) \subseteq$ *synonyms(E).*

Similarly for inheritance, we define $O_\theta : X \rightarrow_\theta Y$, where θ is a user defined threshold representing the allowed path length between two nodes in S. An instance I satisfies O_θ if for every equivalence class $x \in \Pi_X(I)$, all tuples in $\Pi_A(g_x(X))$ are descendants of the same least common ancestor (lca). Furthermore, to identify related values in S, we require the distance between each value in $\Pi_A(g_x(X))$ and the lca to be within a distance of θ.

DEFINITION 2. *A relation I satisfies an inheritance FD* $X \rightarrow_\theta Y$*, if for each attribute* $A \in Y$*, for each* $x \in \Pi_x(I)$ *there exists a class E, such that* $\Pi_A(g_x[X]) \subseteq descendants(E)$*. In addition, for each* $v \in \Pi_A(g_x[X])$*, distance(v, lca)* $\leq \theta$*.*

Related Work. We first discuss the relationship between OFDs and two types of dependencies used in data cleaning (FDs, and metric FDs), and then briefly describe related work. Synonym FDs subsume FDs, since we can create a database where all values have a single string representation, i.e., for all classes E, $|$*synonyms*$(E)|$ = 1. If we set $\theta = 0$, then an inheritance FD becomes a synonym FD, thus, inheritance FDs subsume traditional FDs.

Metric FDs are defined when two tuples agree on X, then the Y values must have similar values w.r.t. some metric distance [9, 10]. OFDs, however, are defined over the values in equivalence classes in Π_{XY}, and there must exist a common class across these values. Consider Table 2, where synonym FD $X \rightarrow_{syn} Y$ is falsified. For each Y value, the defined classes are given in the last column. Although all pairs of $t[Y]$ values share a common class (i.e., {*v,w*}: *D*, {*v,z*}: *C*, {*w,z*}: *F*), the intersection of the classes (for each Π_X value) is empty. Furthermore, ontological similarity is not a metric distance since it does not satisfy the identity of indiscernibles (e.g., for synonyms). Thus, OFDs are not a subclass of metric FDs (and vice versa).

Database applications have used ontologies to provide context in schema integration, querying, and keyword search [2, 8]. However, previous work do not consider the notion of senses to distinguish similar terms under an interpretation. For example, without a sense, the term '`jaguar`' is synonymous with '`Mercedes`' and '`tiger`'. However, an application would have an intended interpretation (sense) for '`jaguar`' either as a vehicle or as an animal. By not considering multiple senses, values are transformed to canonical values, and our problem reduces to regular FD discovery.

3. DISCOVERING OFDS

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*.

Generating Candidates. Let an OFD candidate be of the form $O : X \rightarrow A$. To find a complete, minimal set of OFDs, we test whether a candidate is either a synonym FD or an inheritance FD. Similar to traditional FDs, we normalize all OFDs to a single attribute consequent, i.e., $X \rightarrow A$ for any attribute A. A candidate O is non-redundant if its attribute sets X, A are not supersets of already discovered dependencies. The set of possible antecedent values is the collection of all attribute sets, which can be modelled as a set containment lattice. Each node in the lattice represents an

attribute set and an edge exists between sets X and Y if $X \subset Y$ and Y has exactly one more attribute than X. For a node X, we consider candidate OFDs of the form $(X \setminus A) \rightarrow A$, where $A \in X$. We define k to represent the number of levels in the lattice. A relation with n attributes will generate a $k = n$ level lattice, with $k = 0$ representing the top (root node) level. For brevity, Figure 2 shows the search lattice for 4 of the 5 attributes in Table 1.

After computing the partitions $\Pi_A, \Pi_B \ldots$ for single attributes at level $k = 1$, we efficiently compute the partitions for subsequent levels in linear time by taking the product, i.e., $\Pi_{AB} = \Pi_A \cdot \Pi_B$. Each candidate is evaluated by traversing the lattice in a breadth-first search manner. We consider all X consisting of single attribute sets, followed by all 2-attribute sets, and continue level by level to multi-attribute sets until (potentially) level $k = n$. The algorithm efficiency is based on leveraging the work done in previous levels to prune descendent nodes that are supersets of discovered OFDs. We reduce the search space by avoiding the evaluation of these descendent nodes, thereby saving considerable computation time.

Synonym FDs. Given an ontology S, two concepts c_1 and c_2 in S are synonyms if they are semantically equivalent. Algorithm 1 takes an input relational instance I, ontology S, candidate $O : X \rightarrow_{syn} Y$, and verifies whether O is a qualifying synonym FD, if so, O is added to $synFDs$. We assume that the partitions for each attribute set X, Π_X, and the *canonical names(C)* are given.

Algorithm 1 Discover synonym FDs

INPUT I, S, Π_X, *canonical names()* , synFDs = { }

1: **for** each Π_X **do**
2: **for** each C_A **do**
3: bFoundOFD = computeOverlap(Π_X, C_A)
4: **if** bFoundOFD **then**
5: synFDs += $\{X \rightarrow_{syn} A\}$ where $A \in (R \setminus X)$
6: prune candidates $XY \rightarrow_{syn} A, \forall Y \subseteq (R \setminus X)$ [1]
7: **return** synFDs

computeOverlap(Π_X, C_A)

1: $g_x[X]$ = { $t \mid t \in I$ and $t[X] = x$ }
2: **for** each $x \in \Pi_X(I)$ **do**
3: $t = \Pi_A(g_x[X])$, so $t = (t_1, t_2, \ldots, t_m)$
4: **if** canonical names(t_1) $\cap \ldots \cap$ canonical names(t_m) = $\emptyset$ **then**
5: **return** false
6: **return** true

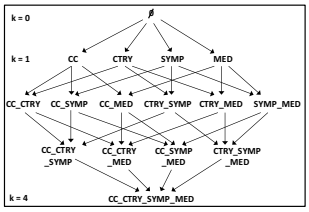

Figure 2: Search lattice.

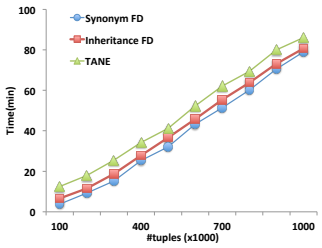

Figure 3: Scalability in N.

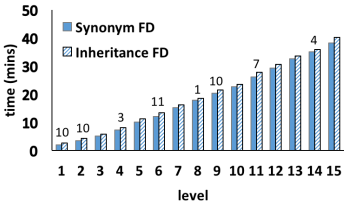

Figure 4: Runtime per level.

[1] We apply further optimizations as described later.

Given a candidate synonym FD $O : X \rightarrow_{syn} A$, we verify whether each value in X corresponds to either equal or synonymous values in A under some common class (sense). If this is true, then O holds over I. In Algorithm 1, for each equivalence class in Π_X, and all tuples within the equivalence class (representing value x), we check whether each corresponding value in A maps to the same canonical value under a common class (sense). If this is true, for all tuples in all equivalence classes, then O holds over I, and supersets of X are pruned. 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 O is then considered in the next $k+1$ level of the search lattice by expanding X with an additional attribute, $((X \cup B) \rightarrow_{syn} A)$.

EXAMPLE 1. *Consider candidate O: [CC]* $\rightarrow_{syn}$ *[CTRY] from Table 1. We have* Π_{CTRY} = *{{*t_1*}{*t_2, t_7*}{*t_3*}{*t_4*}{*t_5*}{*t_6*}} and* Π_{CC} = *{{*t_1, t_5, t_6*}{*t_2, t_4, t_7*}{*t_3*}}. We consider each equivalence class in* Π_{CC}. *The first class, {*t_1, t_5, t_6*}, representing the value 'US' corresponds to three distinct values in CTRY. We check the canonical names('United States'* $\cap$ *'America'* $\cap$ *'USA'). According to a given ontology S, the canonical values resolve to a non-empty value indicating the three references are to the same country. Similarly, the second class {*t_2, t_4, t_7*} corresponds to canonical names('India'* $\cap$ *'Bharat') = 'India'. The last equivalence class {*t_3*} contains a single tuple, thus, there is no conflict. Since we have resolved all references in CTRY to either equal or synonymous values, O holds over Table 1.*

Inheritance FDs. Given an ontology S, two concepts c_1 and c_2 participate in an inheritance relationship if c_2 is a subclass of c_1, or equivalently, c_1 is a superclass of c_2. Given an instance I, ontology S, and candidate $O : X \rightarrow_{\theta} A$, Algorithm 2 determines whether O is an inheritance FD over I if for every equivalence class, the corresponding values in A all share a common ancestor. Furthermore, we consider a restricted version of inheritance that computes the *least common ancestor (LCA)* between all values in $\Pi_A(g_x(X))$, and checks whether the distance between each of the values and the LCA is within a distance θ.

Algorithm 2 Discover inheritance FDs

INPUT I, S, Π_X, θ, *desc.()*

computeLCA(Π_X, C_A)

1: $g_x[X]$ = { $t \mid t \in I$ and $t[X] = x$ }
2: **for** each $x \in \Pi_X(I)$ **do**
3: $t = \Pi_A(g_x[X])$, so $t = (t_1, t_2, \ldots, t_m)$
4: **if** LCA(desc.(canon.nam(t_1)) $\ldots$ desc.(canon.nam(t_m)) $> \theta$) **then**
5: **return** false
6: **return** true

Verifying whether a candidate inheritance FD holds over I proceeds in a similar manner as Algorithm 1. The difference is we replace the computeOverlap() function with computeLCA() to identify values that participate in an inheritance relationship within a distance of θ in S. In Algorithm 2, for each equivalence class in Π_X, we identify the corresponding values in Π_A. For each of these equivalence classes in Π_A, 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 $X \rightarrow_{\theta} A$ holds.

EXAMPLE 2. *Consider candidate* $O : X \rightarrow_{\theta} A$*: [SYMP, DIAG]* $\rightarrow_{\theta}$ *[MED], and the ontology in Figure 1. For each* $x \in \Pi_X$ = *{{*t_1, t_2, t_3*}{*t_4, t_5, t_6*}{*t_7*}}, we identify the corresponding values in* $\Pi_A(g_x(X))$. *For* x_1 = *{*t_1, t_2, t_3*} and* x_4 = *{*t_4, t_5, t_6*}, the LCA is 'analgesic'. Suppose* $\theta = 2$*, all values in* $\Pi_A(g_x(X))$ *are*

within a distance of θ, e.g., distance('ibuprofen', 'analgesic') = distance('tylenol', 'analgesic') = 2. Hence, for θ = 2, O holds.

Foundations and Optimizations. We now present a *sound* and *complete* axiomatization for OFDs. This provides a formal framework for reasoning about OFDs. The axioms provide insight into how OFDs behave, and patterns for how dependencies logically follow from others, that are not easily evident reasoning from first principles. .

The axioms (inference rules) for OFDs are presented in Theorem 4. One of the axioms: Identity, generates *trivial* dependencies, which are always true. We introduce additional inference rules, which follow from axioms, as they will be used throughout in the reminder of the section.

LEMMA 1. $(Reflexivity)$ *If* $Y \subseteq X$, *then* $X \rightarrow Y$.

PROOF. $X \rightarrow X$ holds by Identity axiom. Therefore, it can be inferred by the Decomposition inference rule that $X \rightarrow Y$ holds. □

Union inference rule shows what can be inferred from two or more dependencies which have the same sets on the left side.

LEMMA 2. $(Union)$ *If* $X \rightarrow Y$ *and* $X \rightarrow Z$, *then* $X \rightarrow YZ$.

PROOF. We are given $X \rightarrow Y$ and $X \rightarrow Z$. Hence, the Composition axiom can be used to infer $X \rightarrow YZ$. □

Next, we define the *closure* of a set of attributes X over a set of OFDs M. We use the notation $M \vdash$ to state that $X \rightarrow Y$ is provable with axioms from M.

DEFINITION 3. $(closure)$ *The closure of* X, *denoted as* X^+, *with respect to the set of OFDs* M *is defined as* $X^+ = \{A \mid M \vdash X \rightarrow A\}$.

The important information about closure X^+ is that it can be used to determine whether an OFD follows from set of OFDs M by axioms. The following lemma shows how.

LEMMA 3. $M \vdash X \rightarrow Y$ *iff* $Y \subseteq X^+$.

PROOF. Let $Y = \{A_1, ..., A_n\}$. Assume $Y \subseteq X^+$. By definition of X^+, $X \rightarrow A_i$, for all $i \in \{1, ..., n\}$. Therefore, by Union inference rule, $X \rightarrow Y$ follows. The other direction, suppose $X \rightarrow Y$ follows from the axioms. For each $i \in \{1, ..., n\}$, $X \rightarrow A_i$ follows by the Decomposition axiom. Therefore, $Y \subseteq X^+$. □

THEOREM 4. *These axioms are sound and complete for OFDs.*
1. *Identity:* $\forall X \subseteq R$, $X \rightarrow X$
2. *Decomposition:* $X \rightarrow Y$ *and* $Z \subseteq Y$, *then* $X \rightarrow Z$
3. *Composition: If* $X \rightarrow Y$ *and* $Z \rightarrow W$, *then* $XZ \rightarrow YW$

PROOF. First w prove that the axioms are sound. That is, if $M \vdash X \rightarrow Y$, then $M \models X \rightarrow Y$. The Identity axioms is clearly sound. We cannot have a relation with tuples that agree on X yet are not in synonym or generalization relationship, respectively. To prove Decomposition, suppose we have a relation that satisfies $X \rightarrow Y$ and $Z \subseteq Y$. Therefore, for all tuples that agree on X, they are in synonym or generalization relationship on all attributes in Y and hence, also on Z. Therefore, $X \rightarrow Z$. The soundness of Composition is an extension of the argument given previously.

Below we present the completeness proof, that is, if $M \models X \rightarrow Y$, then $M \vdash X \rightarrow Y$). Without a loss of generality, we consider a table I with three tuples shown in Table 3. We divide the attributes of a relation I into three subsets: X, the set consisting of

X^+		
X	$X^+ \setminus X$	Other attributes
$v...v$	$v...v$	$v...v$
$v...v$	$v'...v'$	$w...w$
$v...v$	$v''...v''$	$u...u$

Table 3: Table template for OFDs.

Patient ID	Country	Country Code	Symptom
10	Canada	CAD	Fever
11	Canada	CA	Congestion
12	Canada	CAD	Pyrexia

Table 4: Table that shows lack of transitivity

attributes in the closure X^+ minus attributes in X and all remaining attributes. Assume that the values v, v' and v'' are not equal ($v \neq v'$, $v \neq v''$ and $v' \neq v''$), however, they are in synonym or generalization relationship, respectively. Also, v, w and u are not in synonym, generalization and component relationship, and hence, they are also not equal.

We first show that all dependencies in the set of OFDs M are satisfied by a table I ($I \models F$). Since OFD axioms are sound, OFDs inferred from M are true. Assume $V \rightarrow Z$ is in M, however, it is not satisfied by a relation I. Therefore, $V \subseteq X$ because otherwise tuples of I disagree on some attribute of V since v, v' and v'' as well as v, w, u are not equal, and consequently an OFD $V \rightarrow Z$ would not be violated. Moreover, Z cannot be a subset of X^r ($Z \not\subseteq X^+$), or else $V \rightarrow Z$, would be satisfied by a table I. Let A be an attribute of Z not in X^+. Since, $V \subseteq X$, $X \rightarrow V$ by Reflexivity. Also a dependency $V \rightarrow Z$ is in M, hence, by Decomposition, $V \rightarrow A$. By Composition $XV \rightarrow VA$ can be inferred, therefore, $X \rightarrow VA$ as $V \subseteq X$. However, then Decomposition rule tells us that $X \rightarrow A$, which would mean by the definition of the closure that A is in X^+, which we assumed not to be the case. Contradiction. An OFD $V \rightarrow Z$ which is in M is satisfied by I.

Our remaining proof obligation is to show that any OFD not inferable from set of OFDs M with OFD axioms ($M \nvdash X \rightarrow Y$) is not true ($M \not\models X \rightarrow Y$). Suppose it is satisfied ($M \models X \rightarrow Y$). By Reflexivity $X \rightarrow X$, therefore, by Lemma 3 $X \subseteq X^+$. Since $X \subseteq X^+$ it follows by the construction of table I that $Y \subseteq X^+$. Else tuples of table I agree on X but are not in synonym, generalization or component relationship, respectively, on some attribute A from Y. Then, from Lemma 3 it can be inferred that $X \rightarrow Y$. Contradiction. Thus, whenever $X \rightarrow Y$ does not follow from M by OFDs axioms, M does not logically imply $X \rightarrow Y$. That is the axiom system is complete over OFDs, which ends the proof of Theorem 4. □

We note that some axioms that hold for FDs do not hold for OFDs, e.g., transitivity if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$. These axioms form the building blocks for our optimizations below.

EXAMPLE 3. *Consider the relation with three tuples in Table 3. The synonym FD* $CTRY \rightarrow_{syn} CC$ *holds since "CAD" and "CA" are synonyms. In addition,* $CC \rightarrow_{syn} SYMP$ *holds as "CAD" and "CA" are not equal, i.e., "CAD" ≠ "CA". However, the transitive synonym FD:* $CC \rightarrow_{syn} SYMP$ *does not hold as "congestion" is not a synonym to both "fever" and "pyrexia".*

LEMMA 5. *If* $A \in X$, *then* $X \rightarrow A$.
PROOF. It follows from Reflexivity (Lemma 1) □

If $A \in X$, then $X \rightarrow A$ is a trivial dependency (Reflexivity).

LEMMA 6. *If $X \rightarrow A$ is satisfied over I, then $XY \rightarrow A$ is satisfied for all $Y \subseteq R \setminus X$.*

PROOF. Assume $X \rightarrow A$. The OFD $X \rightarrow \{\}$ follows from Reflexivity ((Lemma 1)). Hence, it can be inferred by Composition that $XY \rightarrow A$. □

This states that if $X \rightarrow A$ holds in I, then all OFDs with supersets of X on the right-hand-side also hold in I (Augmentation).

LEMMA 7. *If X is a key (or super-key) in I, then for any attribute A, $X \rightarrow A$ is satisfied in I.*

PROOF. Since X is a super-key, partition Π_X consists of singleton equivalence classes only. Hence, the OFD $X \rightarrow A$ is valid. □

If X is a key, then for all $x \in \Pi_X$, $|x| = 1$, and $X \rightarrow A$ holds.

LEMMA 8. *If $|\Pi_A(g_x[X])| = 1$ for an $x \in \Pi_X$, then all values in x map to a unique value in $\Pi_A(g_x[X])$.*

PROOF. Singleton equivalence classes over attribute set X cannot falsify any OFD $X \rightarrow A$. □

If all the values in an equivalence class x correspond to the same value in A, then it is not necessary to check for a synonym or inheritance relationship, since a traditional FD is satisfied in x.

We also provide a linear time inference procedure that determines whether an OFD is logically entailed by a set of OFDs. Our inference procedure can be applied on discovered, and subsequently, user refined OFDs to ensure continued minimality. The discovered dependencies can be manually verified by domain experts, thereby saving valuable time rather than manually defining dependencies from scratch.

Algorithm 3 Inference procedure for OFDs

Input: A set of OFDs M, and a set of attributes X.
Output: The closure of X with respect to M.

```
1: M_unused ← M
2: n ← 0
3: X^n ← X
4: loop
5:     if ∃ V ↦ Z ∈ M_unused and V ⊆ X then
6:         X^{n+1} ← X^n ∪ Z
7:         M_unused ← M_unused \ {V ↦ Z}
8:         n ← n + 1
9:     else
10:        return X^n
11: end loop
```

THEOREM 9. *There exists an inference procedure that correctly computes in linear time closure X^+, $X^+ = \{A \mid M \vdash X \rightarrow A\}$, where M denotes a set of OFDs.*

PROOF. First we show by induction on k that if Z is placed in X^k in Algorithm 3, then Z is in X^+.

Basis: $k = 0$. By Identity axiom $X \rightarrow X$.

Induction: $k > 0$. Assume that X^{k-1} consists only of attributes in X^+. Suppose Z is placed in X^k because $V \rightarrow Z$, and $V \subseteq X$. By Reflexivity $X \rightarrow V$, therefore, by Composition and Decomposition, $X \rightarrow Z$. Thus, Z is in X^+.

Now we prove the opposite, if Z is in X^+, then Z is in the set returned by Algorithm 3. Suppose Z is in X^+ but Z is not in the set returned by Algorithm 3. Consider table I similar to that in Table 3. Table I has three tuples that agree on attributes in X, are in synonym, generalization or component relationship, respectively, but not equal on $\{X^n \setminus X\}$, and are not in synonym, generalization or component relationship, respectively, on all other attributes (hence, also no equal). We claim that I satisfies M. If not, let $P \rightarrow Q$ be a dependency in M that is violated by I. Then $P \subseteq X$ and Q cannot be a subset of X^n, if the violation happens. Similar argument was used in the proof of Theorem 4. Thus, by Algorithm 3, Lines 5–8 there exists X^{n+1}, which is a contradiction. □

EXAMPLE 4. *Let M be the set of generalization FDs from our running example in Table 1: $CC \rightarrow CTRY$ and $\{CC, DIAG\} \rightarrow MED$. Note that generalization FD $CC \rightarrow CTRY$ holds since synonym FD $CC \rightarrow_{syn} CTRY$ and generalization FDs subsume synonym FDs. Therefore, the closure $\{CC, DIAG\}^+$ computed with our inference procedure (Algorithm 9) is $\{CC$, CTRY, $DIAG$, $MED\}$*

For a given set of OFDs M, we can find an equivalent set with a number of useful properties. A minimal set of OFDs is a set with single attributes in the consequence that contain no redundant attributes in the antecedent and that contain no redundant dependencies. We assumed that the input OFDs for our repair algorithm are minimal. To achieve this, we can apply the inference procedure described above to compute a minimal cover of a set of metric FDs.

DEFINITION 4. *A set M of OFDs is minimal if*

1. *$\forall X \rightarrow Y \in M$, Y contains a single attribute;*
2. *For no $X \rightarrow A$ and proper subset Z of X is $M \setminus \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ equivalent to M;*
3. *For no $X \rightarrow Y \in M$ is $M \setminus \{X \rightarrow A\}$ equivalent to M.*

If M is minimal and M is equivalent to a set of metric FDs N, then we say M is a minimal cover of N.

THEOREM 10. *Every set of metric FDs M has a minimal cover.*

PROOF. By the Union and Decomposition inference rules, it is possible to have M with only a single attribute in the right hand side. We can achieve conditions two other conditions by repeatedly deleting an attribute and then repeatedly removing a dependency. We can test whether an attribute B from X is redundant for the OFD $X \rightarrow A$ by checking if A is in $\{X \setminus B\}^+$. We can test whether $X \rightarrow A$ is redundant by computing closure X^+ with respect to $M \setminus \{X \rightarrow A\}$. Therefore, we eventually reach a set of OFDs which is equivalent to M and satisfies conditions 1, 2 and 3. □

EXAMPLE 5. *Assume set of (generalization) OFDs $M = \{M_1 : CC \rightarrow CTRY$, $\{M_2 : CC, DIAG\} \rightarrow MED$, $\{M_3 : CC, DIAG\} \rightarrow \{MED, CTRY\}\}$. Therefore, set of OFDs M is not a minimal cover as M_3 follows from M_1 and M_2 by the Composition axiom.*

Complexity Analysis. The algorithm complexity depends on the number of candidates in the lattice. The worst case complexity is *exponential* in the number of attributes as there are $2^{|n|}$ nodes. However, the complexity is polynomial in the number of tuples. These results are in line with previous FD, inclusion dependency [6], and order dependency [11] discovery algorithms. For OFDs, the ontological relationship (synonym or inheritance) influences the complexity of the verification task. We assume values in the ontology are indexed, and can be accessed within a constant factor. To verify whether a synonym FD holds over I, for each $x \in \Pi_X(I)$, we check whether the intersection of the canonical classes over the consequent values is non-empty. This leads to a worst case time complexity that is quadratic in the number of tuples. A similar argument (checking LCA) applies for an inheritance FD, leading to a worst case complexity that is cubic in the number of tuples.

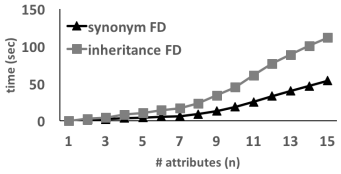

Figure 5: Scalability w.r.t. number of attributes (n).

4. EXPERIMENTS

We evaluate our algorithms using two real datasets (clinical trials from LinkedCT.org of 1M records, and census data from census.gov of 150k records), and refer to the U.S. National Library of Medicine Research [1], and WordNet ontologies.

Exp-1: Performance. We evaluate the scalability of our algorithms against the FD discovery algorithm, TANE [6]. Figure 3 shows a polynomial scale-up with our techniques outperforming TANE by an average of 19.5%. We observed that our clinical trials dataset contained a larger number of satisfying OFDs versus FDs. Hence, while TANE continued to traverse the lower levels of the lattice, our algorithms completed sooner due to the pruning rules. Our optimizations achieve an average 8.6% improvement in running time since we aggressively prune redundant candidates, and avoid visiting the lower levels of the lattice. As expected, we saw an exponential scale-up in running time w.r.t. the number of attributes. Figure 5 show the running times for both discovery algorithms as n increases, with N = 100k, and $\theta = 5$, using the clinical trials dataset. A larger number of inheritance versus synonym FDs are found, leading to the increased running time. We observe that the execution time more than triples from $n = 7$ to $n = 15$, due to the increased number of attribute combinations, and consequently, the larger search space of candidates that must be evaluated.

Exp-2: Efficiency. We argue that compact OFDs (those involving a small number of attributes) are most interesting , and we evaluate the number of OFDs, and the time spent, at each level. OFDs with a larger number of attributes contain more unique equivalence classes. Thus, a less compact dependency may hold over the relation, but not be very meaningful due to overfitting. Figure 4 shows the running time at each level of the lattice, and the total number of OFDs found at each level. Table 5 shows the breakdown for each type of OFD per level (missing levels indicate no OFDs found). For synonym FDs, approximately 61% are found in the first 6 levels taking about 25% of the total time. For inheritance FDs, the results are a bit better, where 63% of the dependencies are found in the first 6 levels taking 16% of the total time. The remaining 35-40% of dependencies (found in the lower levels) are not as compact, and the time to discover these OFDs would take well over 70% of the total time. Since most of the interesting OFDs are found at the top levels, we can prune the lower levels (beyond a threshold) to improve overall running times.

Exp-3: Quality. To measure the accuracy of the discovered OFDs, we used clean versions of the data containing a known number of satisfying synonym and inheritance FDs as the ground truth. We then randomly injected 2% error, and measured the precision and recall. We found the discovered dependencies to be correct, and the recall values ranged between 70%-80%. .

Figure 6 shows the recall results for the clinical trials and census datasets. We found the discovered OFDs to be intuitive and relevant. For example, in the census data, synonym FD O_1: $occupation \rightarrow_{syn} salary$ states that equivalent jobs earn a similar salary. In the clinical data, we found two OFDs, O_2: $[symptoms, diagnosis] \rightarrow_{\theta} medicine$, and O_3 : $[disease] \rightarrow_{syn} medicine$. In O_2, a set of symptoms and diagnosis leads to a prescribed medical treatment,

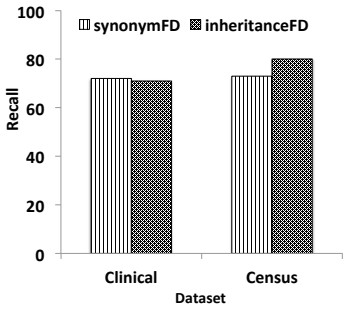

Figure 6: Recall.

level	synFD	inhFD
1	5	5
2	4	6
4	3	-
6	4	7
8	-	1
9	5	5
11	3	4
14	2	2

Table 5: #OFDs found/level.

e.g., a headache diagnosed as a migraine is prescribed ibuprofen for pain relief. In O_3, we found that diseases are treated with similar medicines, but the (medicine) drug names vary across countries. For example, a *fever* is treated with *acetaminophen* in Canada, and with *Paracetamol* in India.

5. CONCLUSIONS

We propose a new class of dependencies, *Ontology Functional Dependencies* that capture domain relationships for data cleaning. Our dependency discovery algorithms identify attribute values satisfying synonym and inheritance relationships in a relation. Our experiments show that our algorithms can be applied to data quality tools that enforce a broad set of attribute relationships w.r.t. an ontology.

As future work, we are considering approximate OFDs, that hold over a portion of the relation. This will enable us to experiment with datasets of varying error rates to achieve greater recall results. We also intend to consider extensions to other relationships such as *component-of* and *type-of*, and the use of ontologies to discover other types of data quality rules such as conditional FDs and denial constraints.

6. REFERENCES

[1] Biomedical terminology, ontologies. *https://mor.nlm.nih.gov*, 2016.

[2] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis. Keyword search over relational databases: A metadata approach. In *SIGMOD*, pages 565–576, 2011.

[3] 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.

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

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

[6] 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.

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

[8] A. Kementsietsidis, L. Lim, and M. Wang. Profile-based retrieval of records in medical databases. In *AMIA*, 2009.

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

[10] N. Prokoshyna, J. Szlichta, F. Chiang, R. Miller, and D. Srivastava. Combining quantitative and logical data cleaning. *PVLDB*, 9(4):300–311, 2015.

[11] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. Effective and complete discovery of order dependencies via set-based axiomatization. *Technical report, CoRR*, abs/1608.06169, 2016.