Contextual Data Cleaning with Ontology Functional Dependencies

ZHENG ZHENG, McMaster University, Canada
LONGTAO ZHENG*, University of Science and Technology of China, China
MORTEZA ALIPOURLANGOURI, McMaster University, Canada
FEI CHIANG, McMaster University, Canada
LUKASZ GOLAB, University of Waterloo, Canada
JAROSLAW SZLICHTA, Ontario Tech University, Canada
SRIDEVI BASKARAN, McMaster University, Canada
ALEXANDER KELLER, Ontario Tech University, Canada

Functional Dependencies (FDs) define attribute relationships based on syntactic equality, and, when used in data cleaning, they erroneously label syntactically different but semantically equivalent values as errors. We explore dependency-based data cleaning with *Ontology Functional Dependencies* (OFDs), which express semantic attribute relationships such as synonyms defined by an ontology. We study the theoretical foundations of OFDs, including sound and complete axioms and a linear-time inference procedure. We then propose an algorithm for discovering OFDs (exact ones and ones that hold with some exceptions) from data that uses the axioms to prune the search space. Towards enabling OFDs as data quality rules in practice, we study the problem of finding minimal repairs to a relation and ontology with respect to a set of OFDs. We demonstrate the effectiveness of our techniques on real datasets, and show that OFDs can significantly reduce the number of false positive errors in data cleaning techniques that rely on traditional FDs.

1 INTRODUCTION

In constraint-based data cleaning, dependencies are used to specify data quality requirements. Data that are 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. Existing approaches use Functional Dependencies (FDs) [8, 38], Inclusion Dependencies [8], Conditional Functional Dependencies [13], Denial Constraints [10], Order Dependencies [44], and Matching Dependencies [14] to define

---

Zheng Zheng, Longtao Zheng, Morteza Alipourlangouri, Fei Chiang, Lukasz Golab, Jaroslaw Szlichta, Sridevi Baskaran, and Alexander Keller

2

Table 1. Sample clinical trials.

| id	CC	CTRY	SYMP	TEST	DIAG	MED
t_1	US	USA	joint pain	CT	osteoarthritis	ibuprofen
t_2	IN	India	joint pain	CT	osteoarthritis	NSAID
t_3	CA	Canada	joint pain	CT	osteoarthritis	naproxen
t_4	IN	Bharat	nausea	EEG	migrane	analgesic
t_5	US	America	nausea	EEG	migrane	tylenol
t_6	US	USA	nausea	EEG	migrane	acetaminophen
t_7	IN	India	chest pain	X-ray	hypertension	morphine
t_8	US	USA	headache	CT	hypertension	cartia
t_9	US	USA	headache	MRI	hypertension	tiazac (ASA)
t_{10}	US	America	headache	MRI	hypertension	tiazac
t_{11}	US	USA	headache	CT	hypertension	tiazac (adizem)
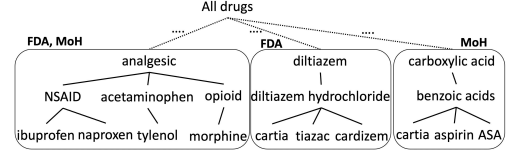
Fig. 1. Medical drug ontology.

the attribute relationships that the data must satisfy. However, these approaches are limited to identifying attribute relationships based on syntactic equivalence (or syntactic similarity for Metric FDs [24, 38]), and unable to convey semantic equivalence, which is often necessary in data cleaning.

EXAMPLE 1.1. *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] → [CTRY] and F_2: [SYMP, DIAG] → [MED]. The tuples ($t_1, t_5, t_6, t_8 - t_{11}$) do not satisfy F_1 as America and USA are not syntactically equivalent (the same is true for (t_2, t_4, t_7)). However, USA is synonymous with America, and ($t_1, t_5, t_6, t_8 - t_{11}$) all refer to the same country. Similarly, Bharat in t_4 is synonymous with India as it is the country's original Sanskrit name. For F_2, ($t_1 - t_3$), ($t_4 - t_6$) and ($t_8 - t_{11}$) are violations as the consequent values all refer to different medications. However, as shown in Figure 1, with domain knowledge from a medical ontology [1], we see that the values participate in an inheritance relationship. Both ibuprofen and naproxen are non-steroidal anti-inflammatory drugs (NSAID), tylenol is an acetaminophen drug, which in turn is an analgesic, and both cartia and tiazac are diltiazem hydrochloride.*

The above example demonstrates that real data contain domain-specific relationships beyond syntactic equivalence or similarity. It also highlights two common relationships that occur 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. Unfortunately, traditional FDs and their extensions 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 reported "errors", and a larger search space of data repairs to consider.

To address these shortcomings, our earlier work proposed a novel class of dependencies called *Ontology Functional Dependencies* (OFDs) that capture synonyms and is-a relationships defined in an ontology [5]. In this paper, we focus on synonyms, which are commonly used in practice, and we study cleaning a relation, and an ontology with respect to (w.r.t.) a set of synonym OFDs. What makes OFDs interesting is the notion of *senses*, which determine how a dependency should be interpreted for a given ontology; e.g., *jaguar* can be interpreted as an *animal* or as a *vehicle*, country codes vary according to multiple standards (interpretations) such as the International Standards Organization (ISO) vs. United Nations (UN). To make OFDs useful in practice, where data semantics are often poorly documented and change, we propose an algorithm to discover OFDs.

OFDs serve as contextual data quality rules that enforce the semantics modeled in an ontology. However, application requirements change, data evolve, and as new knowledge is generated, inconsistencies arise between the data, OFDs, and ontologies. For example, the US Food and Drug Administration (FDA) has a monthly approval cycle to certify new drugs. If downstream data applications do not update their data and ontologies, this leads to stale and missing values in the

ontology, and inconsistencies w.r.t. the data and the OFDs [1]. Similarly, changes to the data may be required to re-align the data to the OFDs and the ontology. Consider the following example.

EXAMPLE 1.2. *Consider Table 1 with updated values (shown in blue) in $t_9[MED]$ and $t_{11}[MED]$ to reflect changes in a patient's prescribed medication. Tuples ($t_8 - t_{11}$) are now inconsistent w.r.t. the previously defined F_2. If we augment F_2 with additional semantics, provided by a medication ontology as shown in Figure 1, tuple t_{11} continues to be problematic since adizem is not defined in the ontology, and is not equivalent to cartia nor tiazac. The updated value in $t_9[MED]$ to ASA leads to an inconsistency since ASA is not semantically equivalent to cartia nor tiazac. We must find an interpretation of the ontology, called a* sense *(denoted in bold in Figure 1), where the values {ASA, cartia, tiazac, adizem} are all equivalent. Unfortunately, there is no sense in which all these values are semantically equivalent. To resolve these violations, we can: (1) repair the ontology by adding the value adizem and ASA under the FDA sense; or (2) update tuples ($t_8 - t_{11}$) to either cartia or ASA under the MoH sense. In both cases, there now exists a sense where all MED values in ($t_8 - t_{11}$) are equivalent.*

The above example demonstrates that there are multiple options to resolve violations, namely, modifying the ontology or the data. We study the repair problem that re-aligns the data, dependencies (OFDs), and an ontology via a minimal number of repairs to the data and to the ontology. State-of-the-art data cleaning solutions have taken a holistic approach to combine attribute relationships with external knowledge bases and statistical methods to determine the most likely repair [38], and probabilistic models that learn from clean distributions in the data to minimally repair dirty values [47]. However, none of these techniques consider repairs to ontologies. We adopt a similar philosophy to prior work that consider repairs to the data and/or constraints [7, 9]. We make the following contributions [1]:

(1) We define OFDs based on synonym relationships. In contrast to existing dependencies, OFDs include attribute relationships that go beyond syntactic equality or similarity, and consider the notion of *senses* that provide the interpretations under which the dependencies are evaluated. In contrast to FDs, OFDs are not amenable to tuple-to-tuple comparisons and instead they must be verified over equivalence classes of tuples.

(2) We introduce a *sound and complete* set of axioms for OFDs. While the inference complexity of other FD extensions is co-NP complete, we show that inference for OFDs remains linear.

(3) We propose the FastOFD algorithm to discover a *complete* and *minimal* set of OFDs to alleviate the burden of specifying them manually. We show that OFDs can be discovered efficiently by traversing a *set-containment lattice* with *exponential* worst-case complexity in the number of attributes, the same as for traditional FDs, and polynomial complexity in the number of tuples. We develop pruning rules based on our axiomatization.

(4) We present OFDClean, an algorithm that computes a minimal number of modifications to an ontology and to the data to satisfy a given set of OFDs. To consider the possible interpretations of the data, OFDClean selects the best interpretation (sense) for an equivalence class of tuples, such that the selected sense minimizes the number of required modifications.

(5) We evaluate the performance and effectiveness of FastOFD and OFDClean using real datasets over varying parameter values.

We give preliminary definitions in Section 2. We present our axiomatization and inference procedure in Section 3, and Section 4 describes our discovery algorithm. We introduce data and ontology repairs, and the OFDClean framework in Section 5. In Section 6, we describe our sense selection algorithm, and then present our ontology repair algorithm in Section 7. We present experimental results in Section 8, related work in Section 9, and conclude in Section 10.

---

[1] We note that the first three contributions are published in our earlier work [5].

2 PRELIMINARIES AND DEFINITIONS

Functional Dependencies. A functional dependency (FD) F over a relation R is represented as $X \rightarrow A$, where X is a set of attributes and A is a single attribute 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 $t_1[A] = t_2[A]$. A partition of X, Π_X, is the set of equivalence classes containing tuples with equal values in X. Let x_i be 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 (in some definitions, we drop the subscript and refer to an equivalence class simply as x). For example, in Table 1, $\Pi_{CC} = \{\{t_1, t_5, t_6\}\{t_2, t_4, t_7\}\{t_3\}\}$.

Sense. An ontology S is an explicit specification of a domain that includes concepts, entities, properties, and relationships among them. In this work, we consider tree-shaped ontologies for simplicity. These constructs are often defined and applicable only under a given interpretation, called a *sense*. The meaning of these constructs for a given S can be modeled according to different senses leading to different ontological interpretations. As mentioned previously, the value *jaguar* can be interpreted under two senses: (1) as an animal, and (2) as a vehicle. As an animal, *jaguar* is synonymous with *panthera onca*, but not with the value *jaguar land rover automotive*.

We define classes E for the interpretations or senses defined in S. Let $synonyms(E)$ be the set of all synonyms for a given class E. For instance, $synonyms(E1) = \{car, auto, vehicle\}$, $synonyms(E2) = \{jaguar, jaguar land rover\}$ and $synonyms(E3) = \{jaguar, panthera onca\}$. Let $names(C)$ be the set of all classes, i.e., interpretations or senses, for a given value C. For example, $names(jaguar) = \{E1, E2, E3\}$ as jaguar can be an animal or a vehicle. Let $descendants(E)$ be a set of all 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* $E\}$, e.g., $descendants(E3) = \{jaguar, peruvian jaguar, mexican jaguar\}$.

Ontology Functional Dependencies We define OFDs w.r.t. a given ontology S. We consider the synonym relationship on the right-hand-side of a dependency leading to synonym OFDs.

DEFINITION 1. *A relation instance I satisfies a* synonym OFD $X \rightarrow_{syn} A$, *if for each equivalence class $x \in \Pi_X(I)$, there exists a sense (interpretation) under which all the A-values of tuples in x are synonyms. That is, $X \rightarrow_{syn} A$ holds, if for each equivalence class x,*

$$\bigcap_{names(a), a \in \{t[A] \mid t \in x\}} \neq \emptyset.$$

EXAMPLE 2.1. *Consider the OFD $[CC] \rightarrow_{syn} [CTRY]$ from Table 1. We have $\Pi_{CC} = \{\{t_1, t_5, t_6\}\{t_2, t_4, t_7\}\{t_3\}\}$. The first equivalence class, $\{t_1, t_5, t_6\}$, representing the value US, corresponds to three distinct values of CTRY. According to a geographical ontology, $names(United States) \cap names(America) \cap names(USA)$ = United States of America. Similarly, the second class $\{t_2, t_4, t_7\}$ gives $names(India) \cap names(Bharat)$ = India. The last equivalence class $\{t_3\}$ contains a single tuple, so there is no conflict. Since all references to CTRY in each equivalence class resolve to some common interpretation, the synonym OFD holds.*

Synonym OFDs subsume traditional FDs, where all values are assumed to have a single literal interpretation (for all classes E, $|synonyms(E)| = 1$). Similar to traditional FDs, we say that an instance I satisfies a set of OFDs Σ, denoted $I \models \Sigma$ *if and only if* I satisfies each OFD $\phi \in \Sigma$. Without loss of generality, we assume that the consequent (right side) of each $\phi \in \Sigma$ consists of a single attribute A (this follows from the axioms that will be presented in Section 3). When we refer to an OFD $\phi: X \rightarrow Y$, we say $I \models \phi$ iff $I \models (X \rightarrow A)$ for all $A \in Y$. That is, for each attribute $A \in Y$, all the A-values of tuples in x share a common sense (interpretation).
Ontological relationships can be semantically meaningful on both sides of a dependency to capture a greater number of true positive errors. However, due to the large search space of ontological relationships on the left-hand side, since we must also consider all tuples outside of the partition

group when checking for violations, we focus on the right-hand-side. This is similar to other seminal work [24, 38] that considers small variations only on the right-hand-side via metric FDs.

Relationship to other dependencies. The notion of senses makes OFDs non-trivial and interesting. For each equivalence class, there must exist a common interpretation of the values. Checking pairs of tuples, as in traditional FDs (and Metric FDs [24, 38], which assert that two tuples whose left-hand side attribute values are equal must have *syntactically* similar right-hand side attribute values according to some distance metric), is not sufficient, as illustrated next.

Consider Table 2, where the synonym OFD $X \rightarrow_{syn} Y$ does not hold (for each Y value, we list its possible interpretations 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 is empty.

Furthermore, OFDs *cannot* be reduced to traditional FDs or Metric FDs. Since values may have multiple senses, it is not possible to create a normalized relation instance by replacing each value with a unique canonical name. Furthermore, ontological similarity is not a metric since it does not satisfy the identity of indiscernibles (e.g., for synonyms).

Table 2. Defining OFDs

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

3 FOUNDATIONS AND OPTIMIZATIONS

In this section, we provide a formal framework for OFDs. We give a sound and complete axiomatization for OFDs that reveals how OFDs behave; notably, not all axioms that hold for traditional FDs carry over. We then use the axioms to design pruning rules that will be used by our OFD discovery algorithm. Finally, we provide a linear time inference procedure that ensures a set of OFDs remains minimal. Due to space constraints, we defer all proofs to the accompanying technical report [49].

3.1 Axiomatization for OFDs

We start with the *closure* of a set of attributes X over a set of OFDs Σ, which will allow us to determine whether additional OFDs follow from Σ by axioms. We use the notation $\Sigma \vdash$ to state that $X \rightarrow Y$ is provable with axioms from Σ.

DEFINITION 2. (*Closure*) *The closure of X, denoted as X^+, with respect to the set of OFDs Σ is defined as $X^+ = \{A \mid \Sigma \vdash X \rightarrow A\}$.*

LEMMA 3.1. $\Sigma \vdash X \rightarrow Y$ *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 the Composition 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 Decomposition, and $Y \subseteq X^+$. □

A sound and complete axiomatization for traditional FDs consists of Transitivity (if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$), Reflexivity (if $Y \subseteq X$ then $X \rightarrow Y$) and Composition. Since OFDs subsume traditional FDs, all lemmas and theorems for OFDs apply to FDs, but not vice versa. For example, Transitivity does not hold for OFDs. Consider a relation $R(A, B, C)$ with three tuples $\{(a, b, d), (a, c, e), (a, b, d)\}$. Assume that b is a synonym of c and d is not a synonym of e. The synonym OFD $A \rightarrow_{syn} B$ holds since b and c are synonyms. In addition, $B \rightarrow_{syn} C$ holds as b and c are not equal. However, the synonym OFD: $A \rightarrow_{syn} C$ does not hold as d and e are not synonyms

Theorem 3.2 below presents a sound and complete set of axioms (inference rules) for OFDs. The Identity axiom generates *trivial* dependencies that are always true.

THEOREM 3.2. *These axioms are sound and complete for OFDs.*
O1 Identity: $\forall X \subseteq R, X \rightarrow X$
O2 Decomposition: If $X \rightarrow Y$ and $Z \subseteq Y$ then $X \rightarrow Z$
O3 Composition: If $X \rightarrow Y$ and $Z \rightarrow W$ then $XZ \rightarrow YW$

Zheng Zheng, Longtao Zheng, Morteza Alipourlangouri, Fei Chiang, Lukasz Golab, Jaroslaw Szlichta, Sridevi Baskaran,
and Alexander Keller

6

PROOF. First we prove that the axioms are sound. That is, if $\Sigma \vdash X \rightarrow Y$ then $\Sigma \models X \rightarrow Y$. The Identity axiom is clearly sound. We cannot have a relation with tuples that agree on X yet are not in a synonym relationship. 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 a synonym relationship on all attributes in Y and hence, also on Z. Therefore, $X \rightarrow Z$. The soundness of Composition is an extension of the same argument.

Below we present the completeness proof, that is, if $\Sigma \models X \rightarrow Y$ then $\Sigma \vdash X \rightarrow Y$. Without loss of generality, we consider a table I with three tuples shown in Table 3. We divide the attributes of I into three subsets: X, the set consisting of attributes in the closure X^+ minus attributes in X, and all remaining attributes. Assume that the values v and v' are not equal ($v \neq v'$ and $v' \neq v''$), but they are in a synonym relationship. Also, v and w are not synonyms, and hence, they are also not equal.

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

Table 3. Table template for OFDs.

We first show that all dependencies in the set of OFDs Σ are satisfied by a table I ($I \models F$). Since OFD axioms are sound, OFDs inferred from Σ are true. Assume $V \rightarrow Z$ is in Σ, however, it is not satisfied by I. Therefore, $V \subseteq X$ because otherwise the tuples of I disagree on some attribute of V since v and v' as well as v and w 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 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 Σ, 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. Our remaining proof obligation is to show that any OFD not inferable from a set of OFDs Σ with OFD axioms ($\Sigma \nvdash X \rightarrow Y$) is not true ($\Sigma \nvDash X \rightarrow Y$). Suppose it is satisfied ($\Sigma \models X \rightarrow Y$). By Reflexivity, $X \rightarrow X$, therefore, by Lemma 3.1, $X \subseteq X^+$. Since $X \subseteq X^+$, it follows by the construction of Table I that $Y \subseteq X^+$. Otherwise, the tuples of Table I agree on X but are not in a synonym relationship on some attribute A from Y. Then, from Lemma 3.1 it can be inferred that $X \rightarrow Y$. Contradiction. Thus, whenever $X \rightarrow Y$ does not follow from Σ by OFDs axioms, Σ does not logically imply $X \rightarrow Y$. That is, the axiom system is complete, and this ends the proof of Theorem 3.2. □

DEFINITION 3. [26] *A functional dependency $X \rightarrow Y$ with* nulls *called Null Functional Dependency (NFD) states that whenever two tuples agree on non-null values in X, they agree on the values in Y, which may be partial.*

THEOREM 3.3. [26] *These axioms are sound and complete for NFDs.*
N1 Reflexivity: $\forall Y \subseteq X, X \rightarrow Y$
N2 Append: If $X \rightarrow Y$ and $Z \subseteq W$, then $XW \rightarrow YZ$
N3 Union: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
N4 Simplification: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Interestingly, the definitions of OFDs and NFDs are semantically different, i.e., a satisfying OFD does not necessarily imply a corresponding NFD is true (e.g., in Table 1 an OFD [CC] $\rightarrow$ [CTRY] holds, but a corresponding NFD [CC] $\rightarrow$ [CTRY] does *not* hold), and vice versa. Also, while data verification for FDs and NFDs can be done on pairs of tuples, for OFDs it has to be performed on an entire equivalence class over the left-hand-side attributes. Following Table 2, 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 these three classes is empty. Hence, $\{t_1, t_2\} \models X \rightarrow Y$, $\{t_2, t_3\} \models X \rightarrow Y$ and $\{t_1, t_3\} \models X \rightarrow Y$, but $\{t_1, t_2, t_3\} \nvDash X \rightarrow Y$. However, their logical inference is equivalent. Despite different bases for OFD and NFD axioms, one can show that the axiom systems are equivalent.

THEOREM 3.4. *The axiom systems for OFDs in Theorem 3.2 and NFDs in Theorem 3.3 are equivalent.*

PROOF. To prove equivalency, we show that all OFD axioms can be proven from NFD axioms and vice versa.

(1) O1.Identity: follows from N1.Reflexivity.
(2) O2.Decomposition: can be inferred from N4.Simplification.
(3) O3.Composition: since $X \to Y$ and $Z \to W$, it follows from N2.Append that $XZ \to Y$ and $XZ \to W$. Hence, by N3.Union, $XZ \to YZ$ is true.

The other direction:

(1) N1.Reflexivity: $X \to X$ follows by O1.Identity, thus, by O2.Decomposition it can be inferred that $X \to Y$.
(2) N2.Append: since $Z \subseteq W$, by O1.Identity and O2.Decomposition, it follows that $W \to Z$. Thus, by O3.Composition $XW \to YZ$.
(3) N3.Union: since $X \to Y$ and $X \to Z$, it follows by O3.Composition that $X \to YZ$.
(4) N4.Simplification: since $Y \subseteq YZ, Z \subseteq YZ$, by O2.Decomposition, we have $X \to Y, X \to Z$.

□

Theorem 3.4 enables us to apply existing algorithms for NFDs to determine whether an OFD holds. Similar conclusions were reached for other classes of dependencies, such as FDs and *pointwise order functional dependencies* (POFDs). While FDs and POFDs are semantically different, the introduced PODs axioms are equivalent to FD axioms, leading to the same inference [33].

Algorithm 1 computes the closure of a set of attributes given a set of OFDs. The inference procedure for NFDs, due to equivalency of inference systems, can be applied to discovered and subsequently user-refined OFDs to ensure continued minimality.

THEOREM 3.5. [26] *Algorithm 1 computes, in linear time, the closure X^+, $X^+ = \{A \mid \Sigma \vdash X \to A\}$, where Σ denotes a set of OFDs.*

For a given set of OFDs Σ, we can find an equivalent *minimal* set, as defined below.

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

(1) $\forall X \to Y \in \Sigma$, Y is a single attribute;
(2) For no $X \to A$ and a proper subset Z of X is $\Sigma \setminus \{X \to A\} \cup \{Z \to A\}$ equivalent to Σ;
(3) For no $X \to Y \in \Sigma$ is $\Sigma \setminus \{X \to A\}$ equivalent to Σ.

If Σ is minimal and equivalent to a set of OFDs Σ', then we say Σ is a minimal cover of Σ'.

EXAMPLE 3.1. *Let $\Sigma = \{\Sigma_1 : CC \to CTRY, \{\Sigma_2 : CC, DIAG\} \to MED, \{\Sigma_3 : CC, DIAG\} \to \{MED, CTRY\}\}$. This set is not a minimal cover as Σ_3 follows from Σ_1 and Σ_2 by Composition.*

3.2 Optimizations

As we will show in Section 4, the search space of potential OFDs is exponential in the number of attributes, as with traditional FDs. To improve the efficiency of OFD discovery, we use axioms to prune redundant and non-minimal OFDs.

LEMMA 3.6. *(Opt-1) If $A \in X$ then $X \to A$.*

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

LEMMA 3.7. *(Opt-2) If $X \to A$ is satisfied over I, then $XY \to A$ is satisfied for all $Y \subseteq R \setminus X$.*

If $X \to A$ holds in I, then all OFDs containing supersets of X also hold in I (Augmentation), and can be pruned. When we identify a key during OFD search, we can apply additional optimizations.

LEMMA 3.8. *(Opt-3) If X is a key (or superkey) in I, then for any A, $X \to A$ is satisfied in I.*

Algorithm 1 Inference procedure for OFDs	**Algorithm 2** FastOFD
Input: A set of OFDs Σ, and a set of attributes X. Output: The closure of X with respect to Σ.	Input: Relation $\mathbf{r}$ over schema $\mathbf{R}$ Output: Minimal set of OFDs Σ, s.t. $\mathbf{r} \models \Sigma$

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

```
 1: L_0 = {}, Σ = {}
 2: C^+({}) = R
 3: l = 1
 4: L_1 = {A | A ∈ R}
 5: while L_l ≠ {} do
 6:    computeOFDs(L_l)
 7:    L_{l+1} = calculateNextLevel(L_l)
 8:    l = l + 1
 9: return Σ
```

For a candidate OFD $X \rightarrow A$, if X is a key, then for all $x \in \Pi_X$, $|x| = 1$, and $X \rightarrow A$ always holds. On the other hand, if $X \setminus A$ is a superkey but not a key, then clearly the OFD $X \rightarrow A$ is not minimal. This is because there exists $B \in X$, such that $X \setminus B$ is a superkey and $X \setminus B \rightarrow A$ holds.

LEMMA 3.9. *(Opt-4) If all tuples in an equivalence class $x \in \Pi_X$ have the same value of A, then a traditional FD, and therefore an OFD, is satisfied in x.*

A *stripped partition* of Π_X, denoted Π_X^*, removes all the equivalence classes of size one. For example, in Table 2, $\Pi_{CC} = \{\{t_1, t_5, t_6\}, \{t_2, t_4, t_7\}, \{t_3\}\}$, whereas the stripped partition removes the singleton equivalence class $\{t_3\}$, so $\Pi_{CC}^* = \{\{t_1, t_5, t_6\}, \{t_2, t_4, t_7\}\}$. If $\Pi_X^* = \{\}$, then X is a superkey and Optimization 3 applies.

LEMMA 3.10. *Singleton equivalence classes over attribute set X cannot violate any OFD $X \rightarrow A$.*

4   OFD DISCOVERY

We now present an algorithm to discover a complete and minimal set of OFDs from data. Based on our axiomatization for OFDs, we normalize all OFDs to a single attribute consequent, i.e., $X \rightarrow A$ for any attribute A. An OFD $X \rightarrow A$ is *trivial* if $A \in X$ by Reflexivity. An OFD $X \rightarrow A$ is *minimal* if it is non-trivial and there is no set of attributes $Y \subset X$ such that $Y \rightarrow A$ holds by Augmentation.

The set of possible antecedent (left side) and consequent (right side) values considered by our algorithm can be modeled 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. Let k be the number of levels in the lattice. A relation with n attributes generates a $k = n$ level lattice, with $k = 0$ representing the top (root node) level.

After computing the stripped partitions $\Pi_A^*, \Pi_B^* \ldots$ for single attributes at level $k = 1$, we compute the stripped partitions for subsequent levels in linear time by taking the product, i.e., $\Pi_{AB}^* = \Pi_A^* \cdot \Pi_B^*$. OFD candidates are considered 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 until (potentially) level $k = n$, similarly as for other use cases of Apriori [4].

Algorithm 2 outlines the OFD discovery process. In level L_l of the lattice, we generate candidate OFDs with l attributes using *computeOFDs*(L_l). FastOFD starts from singleton sets of attributes and works its way to larger attribute sets through the lattice, level by level. When the algorithm processes an attribute set X, it verifies candidate OFDs of the form $(X \setminus A) \rightarrow A$, where $A \in X$. This guarantees that only non-trivial OFDs are considered. For each candidate, we check if it is a valid

---

Algorithm 3 calculateNextLevel($\mathcal{L}_l$)

1: $L_{l+1} = \{\}$
2: **for all** $\{YB, YC\} \in singleAttrDiffBlocks(L_l)$ **do**
3:   $X = Y \bigcup \{B, C\}$
4:   Add X to L_{l+1}
5: **return** L_{l+1}

Algorithm 4 computeOFDs($\mathcal{L}_l$)

1: **for all** $X \in L_l$ **do**
2:   $C^+(X) = \bigcap_{A \in X} C^+(X \setminus A)$
3: **for all** $X \in L_l$ **do**
4:   **for all** $A \in X \cap C^+(X)$ **do**
5:     **if** $X \setminus A \rightarrow A$ **then**
6:       Add $X \setminus A \rightarrow A$ to Σ
7:       Remove A from $C^+(X)$

---

synonym OFD. The small-to-large search strategy guarantees that only minimal OFDs are added to the output set Σ, and is used to prune the search space. The OFD candidates generated in a given level are checked for *minimality* based on the previous levels and are added to a *valid* set of OFDs Σ if applicable. The algorithm *calculateNextLevel(L_l)* forms the next level from the current level. Next, we explain how we check for minimality, and routines *calculateNextLevel* and *computeOFDs*.

4.1 Finding Minimal OFDs

FastOFD traverses the lattice until all minimal OFDs are found. To check if an OFD is minimal, we need to know if $X \setminus A \rightarrow A$ is valid for any $Y \subset X$. If $Y \setminus A \rightarrow A$, then by Augmentation $X \setminus A \rightarrow A$ holds. An OFD $X \rightarrow A$ holds for any relational instance by Reflexivity, therefore, considering only $X \setminus A \rightarrow A$ guarantees that only non-trivial OFDs are taken into account.

We maintain information about minimal OFDs, in the form of $X \setminus A \rightarrow A$, in the *candidate* set $C^+(X)$. If $A \in C^+(X)$ for a given set X, then A has not been found to depend on any proper subset of X. Therefore, to find minimal OFDs, it suffices to verify OFDs $X \setminus A \rightarrow A$, where $A \in X$ and $A \in C^+(X \setminus B)$ for all $B \in X$.

EXAMPLE 4.1. *Assume that $B \rightarrow A$ and that we consider the set $X = \{A, B, C\}$. As $B \rightarrow A$ holds, $A \notin C^+(X \setminus C)$. Hence, the OFD $\{B, C\} \rightarrow A$ is not minimal.*

DEFINITION 5. $C^+(X) = \{A \in \boldsymbol{R} \mid \forall_{A \in X} X \setminus A \rightarrow A \text{ does not hold}\}$.

Some of our techniques are similar to *TANE* [21] for FD discovery and *FASTOD* [43] for Order Dependency (OD) discovery since OFDs subsume FDs and ODs subsume FDs. However, FastOFD differs from *TANE* and *FASTOD* in the optimizations, how nodes are removed from the lattice, and applying keys to prune candidates. FastOFD includes OFD-specific rules. For instance, for FDs if $\{B, C\} \rightarrow A$ and $B \rightarrow C$, then $B \rightarrow A$ holds; hence, $\{B, C\} \rightarrow A$ is non-minimal. However, this rule does not hold for OFDs, and therefore our definition of candidate set $C^+(X)$ differs from *TANE*.

4.2 Computing Levels

Algorithm 3 explains *calculateNextLevel(L_l)*, which computes L_{l+1} from L_l. It uses the subroutine *singleAttrDifferBlocks(L_l)* that partitions L_l into blocks (Line 2). Two sets belong to the same block if they have a common subset Y of length $l - 1$ and differ in only one attribute, A and B, respectively. Therefore, the blocks are not difficult to calculate as sets YA and YB can be expressed as sorted sets of attributes. Other usual use cases of Apriori [4] such as *TANE* [21] use a similar approach.

The level L_{l+1} contains those sets of attributes of size $l + 1$ which have their subsets of size l in L_l.

4.3 Computing Dependencies & Completeness

Algorithm 4, *computeOFDs(L_l)*, adds minimal OFDs from level L_l to Σ, in the form of $X \setminus A \rightarrow A$, where $A \in X$. The following lemma shows that we can use the candidate set $C^+(X)$ to test whether $X \setminus A \rightarrow A$ is minimal.

Lemma 4.1. *An OFD $X \setminus A \to A$, where $A \in X$, is minimal iff $\forall_{B \in X} A \in C^+(X \setminus B)$.*

By Lemma 4.1, the steps in Lines 2, 4, 5 and 6 guarantee that the algorithm adds to Σ only the minimal OFDs of the form $X \setminus A \to A$, where $X \in L_l$ and $A \in X$. In Line 5, to verify whether $X \setminus A \to A$ is a synonym OFD, we apply Definition 1.

The worst case complexity of our algorithm is *exponential* in the number of attributes as there are $2^{|n|}$ nodes in the lattice. The worst-case output size is also exponential in the number of attributes, and occurs when the minimal OFDs are in the widest middle level of the lattice. This means that a polynomial-time discovery algorithm in the number of attributes cannot exist. These results are in line with previous FD [21], inclusion dependency [37], and order dependency [43] discovery algorithms. However, the complexity is polynomial in the number of tuples, although the ontological relationships (synonyms) influence the complexity of verifying whether a candidate OFD holds. We assume that values in the ontology are indexed and can be accessed in constant time.

Ontology FD candidate verification differs from traditional FDs. Following Definition 1 to verify whether a candidate synonym OFD holds over I, for each equivalence class $x \in \Pi_X(I)$, we need to check whether the intersection of the corresponding senses is not empty. This can be done in linear time in the number of tuples by scanning the stripped partitions and maintaining a hash table with the frequency counts of all the senses for each equivalence class. Returning to the example in Table 2, the synonym OFD $X \to_{syn} Y$ does not hold because for the single equivalence class in this example, of size three, there are no senses (classes) for Y that appear three times.

Lemma 4.2. *Let candidates $C^+(Y)$ be correctly computed $\forall Y \in L_{l-1}$. computeOFDs(L_l) calculates correctly $C^+(X)$, $\forall X \in L_l$.*

Theorem 4.3. *The* FastOFD *algorithm computes a complete and minimal set of OFDs Σ.*

5 DATA CLEANING WITH OFDS

Over time, misalignment may arise between a data instance I, an ontology S, and a set of OFDs Σ. In this section, we study the problem of how to compute repairs to re-align I, S with Σ. Data naturally evolve due to updates and changes in domain semantics. These changes in semantics also lead to ontology incompleteness, as new concepts and ontological relationships are introduced. For example, new uses of medical drugs lead to new prescriptions to treat new and different illnesses. While OFDs may also evolve, we argue that repairs to the data I and ontology S are more common in practice, and hence, our current focus. When stale OFDs do occur, FastOFD can be used to discover the latest set of OFDs that hold over the data. For approximate OFDs defined over a dirty instance I, violating values in I can be repaired, thereby transforming approximate OFDs to OFDs that are satisfied over all tuples in I. Formally, we modify I and S to produce I' and S', such that a repaired version of I, $I' \models \Sigma$, w.r.t. a modified version S' of S.

5.1 Scope of Repairs

OFDs interact when two or more dependencies share a common set of attributes, which may invalidate already performed repairs. For simplicity, we assume that there does not exist an attribute that occurs on the left side of one OFD and the right side of another. This simplifies the interaction among repairs to only consequent attributes without having to consider changes to antecedent (left-side) attributes of the dependency, i.e., the equivalence classes for an OFD remain fixed. In many domains, attributes serve as either an independent or dependent role. For example, in health care, demographic and prognosis characteristics are typically independent and dependent attributes, respectively. However, our techniques can handle OFDs that share the same consequent attribute. **Data Repair.** Consider a synonym OFD ϕ: $X \to_{syn} A$, and an equivalence class $x \in \Pi_X(I)$ where there exists at least two tuples $t_1, t_2 \in x$, such that $t_1[A]$ and $t_2[A]$ are not synonyms under some
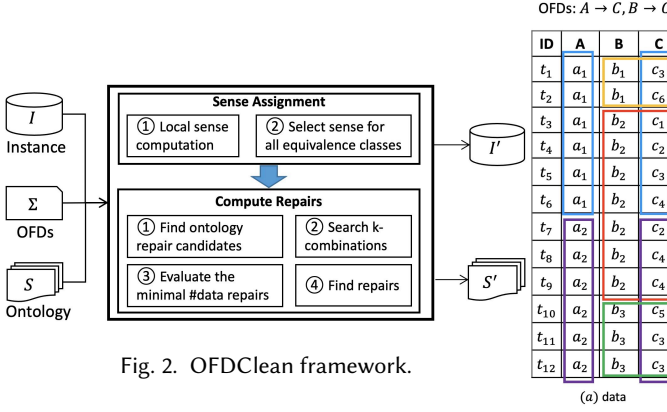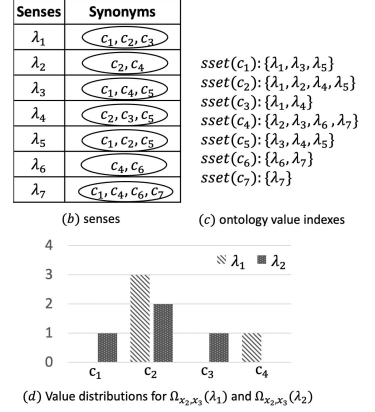
Fig. 2. OFDClean framework.

Fig. 3. Sense assignment.

sense λ. We consider repairs to values in the consequent attributes where the domain of repair values are values in S. Let $\mathcal{P}(I)$ denote the set of all possible data repairs of I. For a repair I' of I, we use $dist(I, I')$ to denote the difference between I and I', which is measured by the number of cells whose values in I are different from the values of the corresponding cells in I'.

Ontology Repair. A repair to an ontology S is the insertion of new value(s) to a node in S w.r.t. a sense λ. Let $\mathcal{P}(S)$ denote the set of possible ontology repairs to S. For a repair S' of S, we define distance $dist(S, S')$ as the number of new values (*concepts*) in S' that are not in S.

Minimal Repair. The universe of repairs, U, represents all possible pairs (S', I') such that $S' \in \mathcal{P}(S)$ and $I' \in \mathcal{P}(I)$ and $I' \models \Sigma$ w.r.t. S'. We want to find minimal repairs in U that do not make unnecessary changes to S or I in a Pareto-optimal sense.

DEFINITION 6. (*Minimal Repair*). *We are given an instance I, OFDs Σ, and an ontology S, where $I \not\models \Sigma$ w.r.t. S. A repair $(S', I') \in U$ is minimal if $\nexists (S'', I'') \in U$ such that $dist(S, S'') < dist(S, S')$ and $dist(I, I'') < dist(I, I')$.*

For example, suppose U contains three repairs: the first one makes two changes each to S and I, the second one makes two changes to S and three changes to I, and the third one makes one change to S and five changes to I. Here, the first and the third repair are minimal.

Problem: *For a data instance I, an ontology S, a set of OFDs Σ, where $I \not\models \Sigma$, compute a repaired instance I', and ontology S', such that $I' \models \Sigma$ w.r.t. S', and $dist(I, I')$ and $dist(S, S')$ are minimal.*

5.2 OFDClean **Overview**

OFDs enrich the semantic expressiveness of FDs via ontological senses, which allow for multiple interpretations of the data. In traditional FDs, senses do not exist, and there is only one interpretation. For a set of OFDs Σ, the problem of finding a minimal repair to S and to I requires evaluating candidate senses for each equivalence class w.r.t. each $\phi \in \Sigma$, and assigning a sense to each class.

Existing constraint-based repair algorithms are insufficient as they do not consider senses [9, 18]. We therefore propose OFDClean, a framework that includes ontology and data repair with sense assignment to minimize the number of changes to I, and S. Intuitively, we consider ontology repairs of size k, where k ranges from one to the total number of ontology repair values. For each candidate ontology repair of size k, we select the ontology repair S' such that $dist(S, S')$ is minimal. For each value k, and the selected S', we compute a repair I' of I such that the number of updates is minimal, and less than a threshold τ, i.e., $dist(I, I') \leq \tau$ to achieve consistency with Σ. We call such repairs

τ-*constrained repairs.* To generate a Pareto-optimal set of repairs, we select (S', I') that are minimal for each k value.

OFDClean takes a greedy, two-step approach. First, we compute an initial sense assignment for each equivalence class that minimizes the number of repairs to the ontology and to the data (locally) w.r.t. each OFD and each equivalence class. Secondly, we go beyond a single OFD to consider pairwise interactions among $\phi \in \Sigma$ where attribute overlap occurs, such that repairing values w.r.t. ϕ_1 may increase or decrease the number of errors in ϕ_2. We model the overlapping tuples between ϕ_1 and ϕ_2 as two distributions w.r.t. their assigned senses. We refine the sense assignment such that the number of necessary updates to transform one distribution to the other is minimized. Lastly, with a computed sense assignment, our problem reduces to finding a minimal repair of S and to I such that it satisfies Σ. Figure 2 provides an overview of OFDClean. Given I, S and Σ, OFDClean identifies inconsistencies in I w.r.t. Σ and S, and returns a set of minimal repairs (I', S') such that $I' \models \Sigma$ w.r.t. S'. OFDClean execution proceeds along two main modules.

Sense Assignment. Let $\phi : X \rightarrow_{syn} A$ be a synonym OFD, where for each equivalence class $x \in \Pi_X$, x is interpreted w.r.t. an assigned sense λ_x. Given I, S, and Σ, for each $\phi \in \Sigma$, we compute λ_x for each $x \in \Pi_X$ such that (I', S') is minimal and $I' \models \phi$. We denote the set of all senses across all x for ϕ as $\Lambda(\phi) = \cup_{x \in \Pi(X)} \lambda_x$. Intuitively, we compute λ_x for every $x \in \Pi_X$ to obtain an interpretation that can achieve a minimal repair. We take a local greedy approach to consider only ϕ when computing $\Lambda(\phi)$. We then model the interactions among the OFDs in Σ, via their overlapping equivalence classes, and refine the initial sense assignment. We quantify interactions and conflicts by comparing the difference between the distributions represented via a pair of equivalence classes. The final assignment of senses for all $\phi \in \Sigma$ is denoted as $\Lambda(\Sigma) = \cup_{\phi} \Lambda(\phi)$. We present our sense assignment algorithm in Section 6.

Compute Repairs. Given $\Lambda(\Sigma)$, Compute Repairs computes the set of minimal repairs by first generating the set of ontology repair candidates that appear in an equivalence class x, but not in S. It then considers combinations of these candidates. We optimize lattice traversal using a beam-search strategy that selects the top-b candidates to expand at each level of the lattice [28]. We consider data updates to the consequent attribute A from the domain of A and synonyms of A in S under the selected sense. We then select candidate repairs that make at most τ updates. We present details of how ontology and data repairs are computed in Section 7.

6 SENSE ASSIGNMENT

Given senses for an ontology S, we compute λ_x for each x and $\phi \in \Sigma$ such that each repair (I', S') is minimal and τ-constrained. A naive approach evaluates all candidate senses against all $x \in \Pi_X$, leading to an exponential number of solutions, which is not feasible in practice.

In this section, we introduce a two-step greedy approach. First, we build an initial solution by considering all senses against each equivalence class x, and selecting the sense that minimizes the number of repairs for each $x \in \Pi_X$ and for each $\phi : X \rightarrow_{syn} A$, leading to a local minimum. This evaluation greedily selects the sense that leads to the fewest repairs for a given x, without considering interactions between the equivalence classes and their senses. In the second step, we consider the interactions among pairs of equivalence classes x, x', where $x \in \Pi_X, x' \in \Pi_{X'}$ for $\phi, \phi' \in \Sigma$, respectively, by performing a local search to improve the initial solution w.r.t. minimizing the number of repairs. Our greedy search selects pairs of equivalence classes where the modeled data distributions share the greatest distance, indicating a larger amount of conflict. We model the interactions among all $\phi, \phi' \in \Sigma$ that share a common consequent attribute by constructing a *dependency graph*, where nodes represent equivalence classes and edges represent a shared consequent attribute. We traverse this dependency graph to refine the initial solution. We evaluate the tradeoff between updating the initial sense assignment and a new assignment to x by estimating

---

Algorithm 5 Initial_Assignment(x, S, Λ)

1: let $sset(v) \leftarrow \emptyset$
2: **for** each $\lambda \in \Lambda$ **do**
3:     **for** each value $v \in \lambda$ **do**
4:         $sset(v) \leftarrow sset(v) \bigcup \lambda$
5: $k' \leftarrow MAD(x)$
6: $potential\_set(x) \leftarrow \emptyset$
7: **while** $potential\_set(x)$ is $\emptyset$ **do**
8:     **for** each top-k' $\{f(v_1), ..., f(v_n)\} \in MAD(x)$ **do**
9:         $\Lambda' \leftarrow sset(v_1) \bigcap ... \bigcap sset(v_n)$
10:         **if** $\Lambda' \neq \emptyset$ **then**
11:             $potential\_set(x) \leftarrow potential\_set(x) \bigcup \Lambda'$
12:     $k' \leftarrow k' - 1$
13: **for** each sense $\lambda \in potential\_set(x)$ **do**
14:     $cover(\lambda) \leftarrow$ the tuple coverage of λ over x
15: $\lambda_x \leftarrow \lambda \in cover(\lambda)$ with maximal tuple coverage
16: **return** λ_x

Algorithm 6 Local_Refinement(G, x, Λ)

1: $u_1 \leftarrow$ BFS(G)
2: **for** each vertex u_2 connected to u_1 **do**
3:     **if** $w(u_1, u_2) > \theta$ **then**
4:         compute $min\{C\}$ repair
5:         **if** sense reassignment for u_2 to λ' **then**
6:             **if** $w'(u_1, u_2) < w(u_1, u_2)$ **then**
7:                 $\Lambda \leftarrow \Lambda(I) \setminus \{u_2, \lambda\}$
8:                 $\Lambda \leftarrow \Lambda(I) \bigcup \{u_2, \lambda'\}$
9: continue until all all u_1 visited
10: **return** Λ

---

the number of necessary data and ontology repairs to achieve alignment between x and x'. We visit nodes x' within a neighbourhood of the current node x as defined via a distance function. We use the *Earth Mover's Distance (EMD)* to quantify the amount of work needed to transform the data distributions modeled in x to the distribution represented in x' [42]. We visit nodes x' where the EMD values between x and x' are largest, indicating regions with the greatest amount of conflict, while avoiding exhaustive enumeration of all candidate assignments. We first describe our greedy algorithm to generate an initial solution, followed by local neighbourhood refinement.

6.1 Computing an Initial Assignment

For $\phi : X \rightarrow_{syn} A, x \in \Pi_X$, we compute λ_x, an initial sense assignment for each x. To compute a minimal repair, we seek senses λ_x containing as many values as possible (maximal overlap) with values in x. Let t_u, v_u represent the number of distinct tuples and values, respectively, that are not covered by λ_x. Let $\vec{r}_{\lambda_x} = (t_u, v_u)$ be a vector representing the number of necessary tuples and values that must be updated in x under sense λ to satisfy ϕ. We seek a sense λ' such that $\vec{r}_{\lambda'_x} \prec \vec{r}_{\lambda_x}, \forall \lambda_x$.

The enumeration of all candidate senses to x is practically inefficient. We take a greedy approach that finds a λ_x with maximal overlap with values in x. We first construct an index of all senses containing each value $v \in x$, denoted by $sset(v)$. For example, Figure 3(c) shows $sset(c_3) = \{\lambda_1, \lambda_4\}$. To maximize the coverage of values, we naturally seek values with maximal frequency. However, given that values in x may contain errors, we seek an ordering of the values in x that is robust to outliers. The *Median Absolute Deviation (MAD)* measure quantifies the variability of a sample, being more resilient to outliers than the standard deviation [41]. Let $f(v)$ denote the frequency of value v. For a univariate dataset $\{f(v_1), f(v_2), \ldots f(v_n)\}$, MAD is defined as the median of the absolute deviations from the median, i.e., MAD(x) = median$(|f(v_i) - f((\bar{v}))|)$, where $f((\bar{v}))$ is the median of all values in x. We sort the values in x according to their decreasing MAD scores. We iteratively search for a sense λ_x containing as many values as possible from x with the highest MAD scores. We use k' as a (decreasing) counter to find such a sense covering k' values in x, where k' is initialized to n, the number of unique values in x. For each value v_i, we compute the intersection of their respective $sset(v_i)$. If there exists a non-empty intersection of senses, the algorithm ends, and returns these senses as an assignment for x, denoted as $potential\_set(x)$. We select the sense λ_x covering the largest number of tuples in x. Algorithm 5 provides the details.

EXAMPLE 6.1. *Figure 3(a) shows a data instance I for two synonym OFDs $\phi_1 : A \rightarrow C$ and $\phi_2 : B \rightarrow C$. Figure 3(b) shows seven senses and the corresponding synonym values for each sense. Let λ_{x_3} represent equivalence class $x_3 = \Pi_{B=b_2}$. Figure 3(c)) shows the index of senses for each value for all x. We compute the MAD for values in x_3, i.e., $c_1 - c_4$, and determine that the ranked ordering is $\{c_4, c_2, c_1, c_3\}$. We intersect the senses in $sset(c_2)$ and $sset(c_4)$, resulting in λ_2 as the initial sense for x_3.*

6.2 Local Refinement

We now discuss how we model interactions between x and x', and address conflicting sense assignments that can lead to an increased number of repairs.

6.2.1 Modeling Interactions. Let $\Omega_{x,x'}(\lambda) = \{t | t \in (x \cap x')\}$ represent the set of overlapping tuples between two equivalence classes x, x' from ϕ, ϕ', respectively, that share the same consequent attribute, interpreted w.r.t. sense λ. To quantify conflict among ϕ, ϕ', we need to interpret tuple values in the intersection of x and x' under the assigned sense. That is, conflicts may arise when there is a difference in the interpretation of the tuple values. Let x_λ represent the set of values in x when interpreted under sense λ. Given synonym relations, assume that for each sense λ, there exists a canonical value representing all equivalent values in λ. Let $D(x_\lambda)$ represent the distribution of values in x_λ, where values in x that are in λ are represented by the canonical value. If x and x' are assigned the same sense λ, then tuples in their intersection, $(\Omega_{x,x'}(\lambda))$, share the same distribution, $D(\Omega_{x,x'}(\lambda))$. When it is clear from the context, we use $\Omega(\lambda)$ to denote overlapping tuples in x and x'.

EXAMPLE 6.2. *Let $x_2 = \Pi_{A=a_1}$ and $x_3 = \Pi_{B=b_2}$, highlighted in blue and red, respectively, in Figure 3(a). If we assign sense λ_1 to both classes, then the tuples in $\Omega_{x_2,x_3}(\lambda_1) = \{t_3, t_4, t_5, t_6\}$ share the same distribution, as shown via light grey bars in Figure 3(d), assuming that c_2 is the canonical value for λ_1.*

The above example demonstrates that when two distributions, $D(\Omega(\lambda))$ and $D(\Omega(\lambda'))$, are equivalent for $\lambda = \lambda'$, the *outliers*, i.e., values not captured by λ, translate into necessary repairs to achieve consistency w.r.t. tuples in Ω. In the above example, c_4 is the outlier, and we can resolve the inconsistency by updating c_4 to c_2 or by adding c_4 to λ_1. In contrast, when $\lambda \neq \lambda'$, tuples in $\Omega(\lambda)$ and $\Omega(\lambda')$ will share different outliers. For example, consider assigning λ_1 to x_2, and λ_2 to x_3. The distributions for $\Omega_{x_2,x_3}(\lambda_1)$ and $\Omega_{x_2,x_3}(\lambda_2)$ are shown via the light grey bars and dark grey bars, respectively, in Figure 3(d). Assuming c_2 is the canonical value for each sense, the outliers are c_1 and c_3 for λ_2, and c_4 for λ_1. Let $\rho_{\Omega,\lambda}$ represent the set of outliers w.r.t. sense λ, i.e., the unique values in $\Omega(\lambda)$ not covered by λ, $\rho_{\Omega,\lambda} = \{v | v \in \Omega_{x,x'}(\lambda), v \notin \lambda\}$. We can resolve these outliers by: (1) updating the sense of x from λ to a new λ'; (2) adding outlier values $\rho_{\Omega,\lambda}$ to S; or (3) performing data repairs to update the values in $\rho_{\Omega,\lambda}$ to values $v \in \lambda$.

For a class x, we seek a new sense assignment $\lambda'' \neq \lambda$ that includes one or more values in $\rho_{\Omega,\lambda}$. This will cause an increase in the number of data or ontology repairs, as this will require sacrificing at least one value $t[A]$ for attribute A that was covered under λ, $t \in \{x \setminus \Omega_{x,x'}(\lambda)\}$. This penalty occurs since if there existed a better sense λ'' that covered all values in $\rho_{\lambda,x}$ then λ'' would have been assigned to x during the initialization step (Section 6.1). Hence, we can have x keep λ or be re-assigned to λ''. However, we must consider the tradeoff of this re-assignment, by incurring a possible penalty (an increased number of repairs) according to the non-overlapping tuples in x. Alternatively, if x keeps λ, then we must update the outlier values via data and/or ontology repairs.

Achieving Alignment. To evaluate the tradeoff of a sense re-assignment, we define a cost function C that sums the number of data and ontology repairs to align $D(\Omega(\lambda))$ and $D(\Omega(\lambda'))$. For ontology repairs, to re-align $D(\Omega(\lambda))$ and $D(\Omega(\lambda'))$, we must add each outlier value in $\rho_{\Omega,\lambda} \cup \rho_{\Omega,\lambda'}$ to ontology S, via senses λ and λ', i.e., $C = |\rho_{\Omega,\lambda}| + |\rho_{\Omega,\lambda'}|$. Since ontology repairs do not lead to
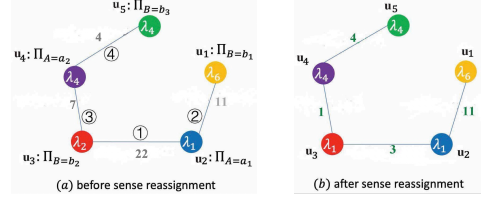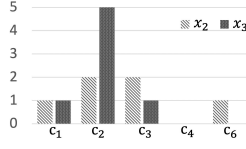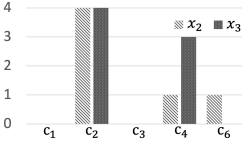
Fig. 4. $x_{2\lambda_1}, x_{3\lambda_1}$ distributions.   Fig. 5. $x_{2\lambda_2}, x_{3\lambda_2}$ distributions.   Fig. 6. Dependency graph.

further conflicts, they incur a cost proportional to the number of values added to λ and S. For data repairs, we resolve differences by updating tuples $\{t|t \in \Omega(\lambda), t[A] \in \rho_{\Omega,\lambda} \cup \rho_{\Omega,\lambda'}\}$ to a value from the candidate set $(\lambda \cap \lambda')$, which guarantees that the repaired value is covered by both senses. We set $C = |\mathcal{R}(\Omega(\lambda))| + |\mathcal{R}(\Omega(\lambda'))|$, where $\mathcal{R}(\Omega(\lambda)) = \{t|t \in \Omega(\lambda), t[A] \in \rho_{\Omega,\lambda}\}$, i.e., the number of tuples containing an outlier value. We select the repair value from the candidate set that minimizes C. Lastly, we consider sense re-assignment that updates λ to λ' for x. For $t \in x$, the delta cost, i.e., the number of data repairs to replace λ with λ' is $C = |\mathcal{R}(x_{\lambda'})| - |\mathcal{R}(x_\lambda)|$, where $\mathcal{R}(x_\lambda) = \{t|t \in x, t[A] \in \rho_{x,\lambda}\}$. For a pair of classes x, x', we consider all three repair options, and select the option that locally minimizes C.

EXAMPLE 6.3. *Figures 4 and 5 show the distribution of values in x_2 and x_3 with initial senses λ_1 and λ_2, respectively. Let c_2 be the canonical value for λ_1 and λ_2. We have: $\rho_{x_2,\lambda_1} = \{c_4, c_6\}$, $\rho_{x_3,\lambda_1} = \{c_4\}$, and $\rho_{x_2,\lambda_2} = \{c_1, c_3, c_6\}$, $\rho_{x_3,\lambda_2} = \{c_1, c_3\}$. We evaluate ontology and data repair, and two sense assignment options: (i) add c_4 to λ_1, and add c_1 and c_3 to λ_2 to eliminate the conflict, leading to $C = 3$; (ii) update three tuples with values $\{c_4, c_1 c_3\}$ to c_2, giving $C = 3$; (iii) assign λ_1 to x_3 (from λ_2), leading to $|\mathcal{R}(x_{3,\lambda_1})| - |\mathcal{R}(x_{3,\lambda_2})| = 3 - 2 = 1$; (iv) assign λ_2 to x_2 where we update $\{c_1, c_3, c_6\}$ to c_2 giving $C = |\mathcal{R}(x_{2,\lambda_2})| - |\mathcal{R}(x_{2,\lambda_1})| = 4 - 2 = 2$. Since C is minimal with option (iii), we re-assign λ_1 to class x_3.*

6.2.2 Identifying Candidate Classes.

In this section, we discuss how to select pairs of classes x, x' for refinement. Visiting all pairs is not feasible, as we must evaluate m choose 2 pairs of classes, where m equals the total number of equivalence classes across all $\phi \in \Sigma$. For classes $x_\lambda, x'_{\lambda'}$, we quantify the deviation between their respective distributions, by measuring the amount of work needed to transform $D(\Omega(\lambda))$ to $D(\Omega(\lambda'))$ (or vice-versa). We quantify this work using the Earth Mover's Distance measure, denoted *EMD* [39].

Let P and Q represent distributions $D(\Omega_{x_1,x_2}(\lambda_1)), D(\Omega_{x_1,x_2}(\lambda_2))$, respectively. We compute $EMD(P,Q)$ and check whether $EMD(P,Q) > \theta$, for a user-given threshold θ. Intuitively, if P and Q are sufficiently different (according to θ), then classes x_1, x_2 are candidates to consider sense re-assignment to re-align P and Q. Towards finding the sense with minimal repair, if the cost C of sense re-assignment and the EMD values are lower with the new sense, we proceed with sense re-assignment for x_1, x_2. Otherwise, we keep the initial senses for x_1 and x_2.

Dependency Graph. Let $G = (V, E)$ denote a *dependency graph*, where each vertex $u \in V$ represents an equivalence class x, and an edge $e = (u_1, u_2) \in E$ exists when the corresponding classes $(x_1 \cap x_2) \neq \emptyset$. Recall that we define overlap between two classes x_1 and x_2 when their respective OFDs share a common consequent attribute. Hence, we only include nodes (and edges) in the dependency graph for OFDs with a common consequent attribute, thereby avoiding enumeration of all pairs of classes. We use the terms u_1, u_2 to represent classes when it is clear from the context. Let $w(u_1, u_2)$ denote the weight of edge e, representing the EMD value between u_1 and u_2. We visit nodes in a breadth-first search manner, and select nodes u_1 with the largest EMD value by summing over all edges containing u_1. This strategy prioritizes classes that require the largest amount of work. For each visited edge, we check whether $w(u_1, u_2) > \theta$, and if so, check whether a sense re-assignment will reduce the weight, i.e., reduce the EMD value under the new sense. If so, then

Algorithm 7 Sense_Assignment(I, Σ, S, θ)

1: $\Lambda \leftarrow \{\emptyset, \emptyset\}$
2: **for** each $\phi \in \Sigma$ **do**
3:      **for** each $x \in \Pi_X(I)$ **do**
4:          **for** each $\lambda \in \Lambda$ **do**
5:              $\lambda_x \leftarrow$ Initial_Assignment(x, S, Λ)
6:              $\Lambda \leftarrow \Lambda \cup \{x, \lambda_x\}$
7: $G =$ CreateDepGraph(I, Σ, S, Λ)
8: **sort**(G, DSC) Sort vertices in DSC order EMD edge weights
9: **while** $(u_1 = BFS(G)$ and $EMD(u_1) > \theta)$ **do**
10:      $\Lambda \leftarrow$ Local_Refinement(G, x, Λ)
11: **return** Λ

Algorithm 8 Ontology_Repair (I, Σ, S, b)

1: $Cand(S) \leftarrow \emptyset$
2: $\mathcal{P}(S) \leftarrow \{\emptyset, ..., \emptyset\}$
3: **for** each $\phi \in \Sigma$ **do**
4:      **for** each $x \in \Pi_X(I)$ **do**
5:          **if** $t_e \in x$ and $t_e[A] \notin S$ **then**
6:              $Cand(S) \leftarrow Cand(S) \cup t_e[A]$
7: $k \leftarrow 1$
8: Generate $\mathcal{V}_k$ as clusters of size k, with values $v \in Cand(S)$
9: $L_k \leftarrow \{v_k | v_k \in \mathcal{V}_k\}$
10: **while** $L_k \neq \emptyset$ **do**
11:      $\delta(v_k) =$ compute data repairs for v_k
12:      $L_k \leftarrow$ top-b nodes with smallest $\delta(v_k)$ to expand
13:      $v_k* = \{v_k | \min \delta(v_k)\}$
14:      $\mathcal{P}(S) \leftarrow (S', I')$ using v_k*
15:      $k \leftarrow k + 1$
16: **return** $\mathcal{P}(S)$

the new sense re-aligns the two class distributions and incurs a minimal cost C. The algorithm ends when all vertices have been visited. Algorithm 6 provides the details.

EXAMPLE 6.4. *Figure 6(a) shows the dependency graph corresponding to the equivalence classes in Figure 3(a), with EMD values as edge weights. Suppose $\theta = 10$, and we visit nodes in BFS order of $\{u_2, u_1, u_3, u_4, u_5\}$. Starting at node u_2 (the blue node), we evaluate edge (u_2, u_3) with weight 22. From Example 6.3, we update the sense for u_3 from λ_2 to λ_1 since the new weight $w'(u_3, u_2) < w(u_3, u_2) = 3 < 22$. We next consider edge (u_2, u_1) with $w(u_2, u_1) = 11$, with costs to update u_1 for data repair, ontology repair and sense reassignment as 1, 1 and 0, respectively. However, after updating u_1 sense λ_6 to λ_1, $w'(u_2, u_1)$ does not decrease. Thus, we do not refine the sense for u_1, and keep λ_6 as is. The algorithm then visits node u_3 and evaluates edge (u_3, u_4), where after sense reassignment to λ_1 for u_3, we have $w(u_3, u_4) = 1 < \theta$, and there is no further evaluation needed. Lastly, we visit nodes u_4, u_5, where $w(u_4, u_5) = 4 < \theta$, and the algorithm terminates. Figure 6(b) shows the final sense assignments.*

6.3 Algorithm

Algorithm 7 presents the details of our sense assignment algorithm. We first compute an initial assignment for every equivalence class x. We construct the dependency graph G, and compute the *EMD* between overlapping classes (u_1, u_2) as edge weights. We visit nodes in decreasing order of their *EMD* values by summing over all corresponding edges. For example, Figure 3(d) shows that we visit u_2 first since the EMD value of $u_2 = 22 + 11 = 33$, and of $u_3 = 22 + 7 = 29$. Furthermore, we visit edge (u_2, u_3) first since $w(u_2, u_3) = 22$ is the largest edge weight of u_2. We refine the sense for each equivalence class based on Algorithm 6. Lastly, the algorithm returns the final sense assignment. **Complexity.** Computing an initial assignment takes time $O(|\Pi_X| \cdot m)$ to evaluate all $|\Pi_X|$ equivalence classes, and in the worst case, the total m senses. Traversing the dependency graph is in the same order as BFS, taking time $O((|V| + |E|) \cdot m)$, and to evaluate senses for each visited node.

7 COMPUTING REPAIRS

After each equivalence class x is assigned a sense, we have an interpretation from which to identify errors, and to compute repairs. We describe our repair algorithm that first evaluates ontology repair

Table 4. Sample clinical trials data.

id	CC	CTRY	SYMP	DIAG	MED
t_8	US	USA	headache	hypertension	cartia
t_9	US	USA	headache	hypertension	ASA
t_{10}	US	America	headache	hypertension	tiazac
t_{11}	US	Uni. States	headache	hypertension	adizem

$label_1$: SYMP,DIAG→MED, adizem→MED (FDA)
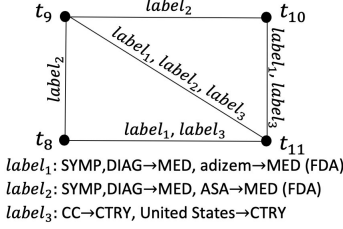$label_2$: SYMP,DIAG→MED, ASA→MED (FDA)
$label_3$: CC→CTRY, United States→CTRY

Fig. 7. Conflict graph for Table 4.

Table 5. Sample repairs.

Ont. Repair value (sense)	$|S, S'|$	Conflict Edges	C_{2opt}	δ_P
∅	0	(t_8, t_9), (t_8, t_{11}), (t_9, t_{10}) (t_9, t_{11}), (t_{10}, t_{11})	t_9, t_{11}	4
ASA (FDA)	1	(t_8, t_{11}), (t_9, t_{11}), (t_{10}, t_{11})	t_{11}	2
adizem (FDA)	1	(t_8, t_9), (t_8, t_{11}), (t_9, t_{10}) (t_9, t_{11}), (t_{10}, t_{11})	t_9, t_{11}	4
United States	1	(t_8, t_9), (t_8, t_{11}), (t_9, t_{10}) (t_9, t_{11}), (t_{10}, t_{11})	t_9, t_{11}	4
adizem (FDA) United States	2	(t_8, t_9), (t_9, t_{10}), (t_9, t_{11})	t_9	2
...	...	...	...	...

candidates, i.e., values that are in I but not in S under the chosen sense. Second, assuming a set of ontology repairs, we discuss how the remaining data violations are modeled and repaired.

7.1 Ontology Repairs

We consider ontology repairs that add new values $v \in I$, but $v \notin S$. Let $Cand(S)$ represent all candidate ontology repair values. We iterate through $Cand(S)$ by considering repairs of size k, $k = \{1, 2, ..., |Cand(S)|\}$. For ease of presentation, let $\mathcal{V}_k$ represent the set of candidate ontology repairs of size k. For each $\mathcal{V}_k$, we compute the number of data repairs δ_k needed to achieve consistency w.r.t. Σ, and select data repairs with a minimum number of updates for each k. To generate a Pareto-optimal set of repairs, we select (S', I') that are minimal for each k value.

Generating all k-combinations of solutions has exponential complexity, taking $O(2^{|Cand(S)|})$. We propose a greedy strategy to selectively consider the most promising solutions. We use the *beam search* strategy, a heuristic optimization of breadth-first search, that expands the top-b most promising nodes, for a beam size b. The parameter b can be tuned according to application requirements; in our case, we select b by maximizing the probability of selecting the value from a random sequence. The *Secretary Problem* addresses this issue, where the objective is to select one secretary from w candidates, and applicants are considered in some random order (all orders are equally likely) [15]. Applicants are chosen immediately after being interviewed, and decisions are non-reversible. Prior work has shown that it is optimal to interview $b = \lfloor \frac{w}{e} \rfloor$ candidates, where e is the exponential constant. This strategy selects the best candidate with probability $\frac{1}{e}$, with a competitive ratio e. In our case, we set $w = |Cand(S)|$.

We organize the candidate repairs as a set-containment lattice, where nodes at level k represent ontology repairs of size k, e.g., at level 1, we consider single value repairs. Let v_k represent a node at level k, i.e., the repairs in $v_k \in \mathcal{V}_k$. Each node v_k is created by augmenting a node at level $k - 1$ with a single repair value. We begin the search at a node v_1 at level $k = 1$. As we visit a node v_k, we compute the minimum number of data repairs for the current candidate ontology repair v_k, i.e., assuming $S' = \{S \cup v_k\}$. We select the top-b nodes at each level k with the minimum number of data repairs, to further explore at the next level $k + 1$. We continue this traversal at each subsequent level until we have $I' \models \Sigma$ w.r.t. S', or until we reach the leaf level. Algorithm 8 presents the details.

7.2 Approximating Minimum Data Repairs

Given v_k to derive S', we compute the necessary data repairs I' such that $dist(I, I') \leq \tau$. Computing a minimal number of data repairs to I such that $I' \models \Gamma$, for a set of FDs Γ, is known to be NP-hard [23]. Since OFDs subsume FDs, i,e., $\Gamma \subseteq \Sigma$, this intractability result, unfortunately, carries over to OFDs. Beskales et al., show that the minimum number of data repairs can be approximated by upper bounding the minimum number of necessary cell changes [7]. In our implementation, we adapt their RepairData algorithm that is shown to compute an instance I', by cleaning tuples one at a time, such that the number of changed cells is at most $P = 2 \cdot \min\{|Z|, |\Sigma|\}$, where $|Z|$ is the number of (unique) consequent attributes in Σ, and $|\Sigma|$ denotes the number of OFDs. We seek P-approximate, τ-constrained repairs, where a P-approximate τ-constrained repair (S', I') is a repair in U such that $dist(I, I') \leq \tau$, and there is no other repair $(S'', I'') \in U$ such that $(dist(S, S''), P \cdot dist(I, I'')) \prec (dist(S, S'), \tau)$.

We transform the data repair problem to the problem of finding a minimum vertex cover, where nodes represent a tuple t_i, and an edge (t_i, t_j) represent that t_i, t_j conflict w.r.t. an OFD. We generate a conflict graph for I, w.r.t. Σ, S', and compute a 2-approximate minimum vertex cover of the conflict graph [17]. We annotate the edges in the conflict graph with: (i) the violated OFD ϕ; and (ii) the candidate repair and sense.

EXAMPLE 7.1. *Table 4 shows a subset of Table 1 where $t_{11}[CTRY]$ is updated to United States. Consider $\phi_1 : [CC] \rightarrow [CTRY]$ with an ontological equivalence between USA, America, and $\phi_2 : [SYMP, DIAG] \rightarrow [MED]$ w.r.t. the drug (MED) ontology shown in Figure 1. If the FDA sense is selected for ϕ_2 (according to Algorithm 7), then Figure 7 shows the corresponding conflict graph.*

For each error tuple t_e, we modify attribute A by considering candidate repairs from the domain of A when there is sufficient evidence to do so, or to a value $v' \in S'$ such that the values $\{v', a\}$ are synonyms for all $a \in \Pi_X$. As each t_e is repaired, we remove its corresponding nodes and edges, and re-generate the conflict graph in linear time. The algorithm continues until all tuples t_e have been removed from the conflict graph, and I' is returned. The algorithm runs in $O(|E| + |I|)$, where $|E|$ is the number of edges in the conflict graph [7].

EXAMPLE 7.2. *Table 5 shows an execution of OFDClean listing: (i) candidate ontology repairs (value, sense); (ii) number of ontology repairs $|\Delta(S, S')|$; (iii) conflicts in the conflict graph (Figure 7); (iv) C_{2opt}: the 2-approximation vertex cover of the conflict graph, indicating the tuples to be repaired; and (v) $\delta_P(S', I)$: the upper bound on the minimum number of data repairs to I. Consider an ontology repair to add ASA to S, which would eliminate edges (t_8, t_9), (t_9, t_{10}), and leave only tuple t_{11} to be updated with at most two minimum data repairs. Comparing all single candidate ontology repairs, this repair is the minimum, involving one and two, ontology and data repairs, respectively.*

8 EXPERIMENTS

Our evaluation focuses on four objectives: (1) We test FastOFD scalability and performance compared to seven FD discovery algorithms as we scale the number of tuples and the number of attributes. (2) We show the benefits of our optimization techniques to prune redundant OFD candidates. (3) We conduct a qualitative evaluation of the utility of the discovered OFDs. (4) We study the accuracy and performance of OFDClean. We show the effectiveness of our sense selection algorithm, and compare OFDClean against the HoloClean[40] repair algorithm and demonstrate the benefits of ontology repair.

Table 6. Parameter values (defaults in bold)

Symbol	Description	Values
$|\lambda|$	# senses	2, **4**, 6, 8, 10
$err\%$	error rate	**3**, 6, 9, 12, 15
N	# tuples (Million)	0.2, **0.4**, 0.6 0.8, 1 (clinical)
b	beam size	1, 2, **3**, 4, 5
$inc\%$	incompleteness rate	2, **4**, 6, 8, 10
$|\Sigma|$	# OFDs	**10**, 20, 30, 40, 50

Table 7. Varying N runtimes.

N (M)	0.2	0.4	0.6	0.8	1
(sec)	9.3	11.8	17.1	23.8	27.2

Table 8. OFDClean runtimes (m)

N (K)	50	100	150	200	250
	166	175	182	198	217

8.1 Experimental Setup

Experiments were performed using four Intel Xeon processors at 3.0GHz each with 32GB of memory. Algorithms were implemented in Java. The reported runtimes are averaged over six executions.

Datasets. We use two real datasets, and the U.S. National Library of Medicine Research and WordNet ontologies [3].

Kiva [2] describes loans issued over two years via the Kiva.org online crowdfunding platform to financially excluded citizens around the world. There are 670K records and 15 attributes, including loan principal amount, loan activity, country code, country, region, funded time and usage.

Clinical [20]: The Linked Clinical Trials (LinkedCT.org) database describes clinical patient data characteristics such as the clinical study, country, medical diagnosis, drugs, illnesses, symptoms, treatment, and outcomes. We use a portion of the dataset with 250K records and 15 attributes.

We use real ontologies to ensure coverage of the (RHS) attribute domain, e.g., the Medical Research ontology covers values in the DRUG attribute of the clinical trials data, while considering some medications have different names across different countries (senses). We aim to maximize coverage (upwards of 90%+ coverage for some attributes). Values not covered by an ontology are candidates for ontology repair.

Parameters and Ground Truth. Table 6 shows the parameter values and their defaults. We inject errors randomly into the consequent attributes. Errors are inserted by either changing an existing value to a new value (not in the attribute domain), or to an existing domain value. We consider the original data values as the ground truth, as we vary the error rate $err\%$. We specify τ as a percentage (100% allows the algorithm to freely change the data). We set $\tau = 65\%$ to balance similarity between I and I', and flexibility to consider new values via ontology repairs.

Comparative Techniques.

FD Mining: For the FastOFD comparative experiments, we use the Metanome implementations of existing FD discovery algorithms [36].

HoloClean [40] provides holistic data repair by combining multiple input signals (integrity constraints, external dictionaries, and statistical profiling), and uses probabilistic inference to infer dirty values. We input to HoloClean: (i) a set of denial constraints translated from the given dependencies; (ii) external reference sources such the National Drug Code Directory [45]; and (iii) we profile the data to obtain statistical frequency distributions of each attribute's domain. We compare OFDClean against HoloClean since it also considers external information during data repair.

8.2 FastOFD Efficiency

Exp-1: Vary $|N|$. We vary the number of tuples, and compare OFDClean (with optimizations) against seven existing FD discovery algorithms: TANE [21], FUN [34], FDMine [48], DFD [50], DepMiner [27], FastFDs [46], and FDep [16]. Figure 8a shows the running times using $\theta = 5$. We report partial results for FDMine and FDep as both techniques exceeded the main memory limits. FDMine returns a much larger number of non-minimal dependencies, about 24x leading to increased

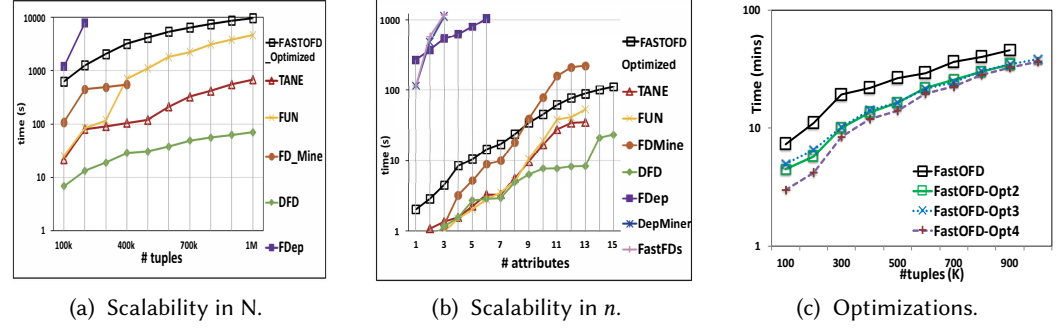(a) Scalability in N.   (b) Scalability in n.   (c) Optimizations.

Fig. 8. FastOFD effectiveness and efficiency (Clinical dataset).

memory requirements. We ran DepMiner and FastFDs using 100K records and report running times of 4hrs and 2.3 hrs, respectively. However, for larger data sizes (200K+ records), we terminated runs for these two techniques after 12 hours. The running times for FastOFD scale linearly with the number of tuples, similar to other lattice traversal based approaches (TANE, FUN, and DFD). The runtime is dominated by data verification of OFDs. Discovering OFDs incurs an increased runtime, i.e., we found that discovering synonym OFDs incurs an average overhead of 1.8x over existing lattice-traversal FD discovery algorithms. This can be explained by the inherent complexity of OFDs (which subsume FDs), and the increased number of discovered OFDs. For example, in the clinical data, many illnesses and medications are referenced by multiple names but refer to the same entity; e.g., a drug can be referred to by its generic name or its brand name.

Exp-2: Vary $|n|$. Figure 8b shows that as we vary the number of attributes (n), using $N = 100k$ tuples, and $\theta = 5$, all algorithms scale exponentially with the number of attributes since the space of candidates grows with the number of attribute set combinations. FastOFD scales comparatively with other lattice based approaches. We discover 3.1x more dependencies on average, compared to existing approaches, validating the overhead we incur. In Figure 8b, we report partial results for DepMiner, FastFDs, and FDep (before memory limits were exceeded), where we achieve almost two orders of magnitude improvement due to our optimizations. Our techniques performs well on a smaller number of attributes due to effective pruning strategies that reduce the search space.

Exp-3: Optimizations Benefits. We evaluate optimizations 2, 3, and 4 (Opt-2, Opt-3, Opt-4) from Section 3.2. We use 1M tuples from the clinical trials dataset. Figure 8c shows FastOFD runtime with each of the optimizations, shown in log-scale. Opt-2 achieves 31% runtime improvement due to aggressive pruning of redundant candidates at lower lattice levels. For Opt-3, we see a smaller performance improvement (compared to Opt-2) since we reduce the verification time due to the existence of keys, rather than pruning candidates. We found two key attributes in the clinical data: (1) NCTID, representing the clinical trials.gov unique identifier for a clinical study; and (2) OrgStudyID, an identifier for a group of related studies. Opt-3 achieves an average 14% runtime improvement. With more keys, we expect to see a greater improvement in running time. For Opt-4, we use a set of five defined FDs [5]. At 100K tuples, we reduce running times by 59%. The running times decrease by an average of 27%. We expect that the performance gains are proportional to the number of satisfying FDs in the data; an increased number of satisfying FDs lead to less time spent verifying candidate OFDs. Our optimizations together achieve an average of 24% improvement in runtime, and our canonical representations are effective in avoiding redundancy.

Exp-4: Efficiency over lattice levels. We argue that compact OFDs (those involving a small number of attributes) are the most interesting. OFDs with more attributes contain more unique equivalence classes. Thus, a less compact dependency may hold, but may not be very meaningful due

(a) Accuracy vs. $|\lambda|$ (b) Time vs. $|\lambda|$ (c) Accuracy vs. $err\%$ (d) Time vs. $err\%$
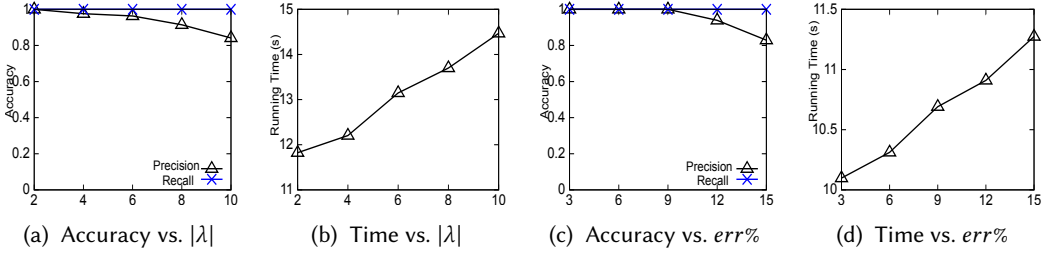
Fig. 9. Sense assignment effectiveness (Clinical dataset).

to overfitting. We evaluate the efficiency of our techniques to discover these compact dependencies. To do this, we measure the number of OFDs and the time spent at each level of the lattice using the clinical trials data. OFDs discovered at the upper levels of the lattice (involving fewer attributes) are more desirable than those discovered at the lower levels. Approximately 61% are found in the first 6 levels (out of 15 levels) taking about 25% of the total time. The remaining 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.

8.3 FastOFD **Effectiveness**

Exp-5: Eliminating False-Positive Data Quality Errors. For each discovered OFD, we sample and compute the percentage of tuples containing syntactically non-equal values in the consequent to quantify the benefits of OFDs versus traditional FDs. These values represent synonyms. Under FD based data cleaning, these tuples would be considered errors. However, by using OFDs, these false positive errors are saved since they are not true errors. For brevity, we report results here, and refer the reader to [5] for graphs. Overall, we observe that a significantly large percentage of tuples are falsely considered as errors under a traditional FD based data cleaning model. For example, at level 1, 75% of the values in synonym OFDs contain non-equal values. By correctly recognizing these 'erroneous' tuples as clean, we reduce computation and the manual burden for users to decipher through these falsely categorized errors.

8.4 **Sense Selection Performance**

We now use the clinical dataset to evaluate the performance of our sense selection algorithm.
Exp-6: Vary $|\lambda|$**.** Figure 9a shows the accuracy as we increase the number of senses $|\lambda|$. Since every equivalence class is assigned a sense, we achieve 100% recall, independent of the number of senses. Precision decreases with more senses due to more available choices, but still remains above 80%. As we evaluate more senses, the runtimes increase linearly, as shown in Figure 9b.
Exp-7: Vary $err\%$**.** Figure 9c shows that precision declines linearly as the number of errors increases. It becomes more challenging to select the correct sense when multiple senses contain overlapping (erroneous) values. Figure 9d shows that runtimes increase as more candidate senses and refinements have to be evaluated to align distributional deviations caused by an increasing number of errors.
Exp-8: Vary N**.** We increase the number of tuples in the Clinical dataset up to 1M records. This increases the number of equivalence classes, but our sense assignment strategy achieves over 90% precision. We found that the increase in N did not impact the precision and recall accuracy (figure omitted for brevity). With more equivalence classes, there is an increased likelihood of overlap, leading to longer runtimes to resolve shared errors, as shown in Table 7.
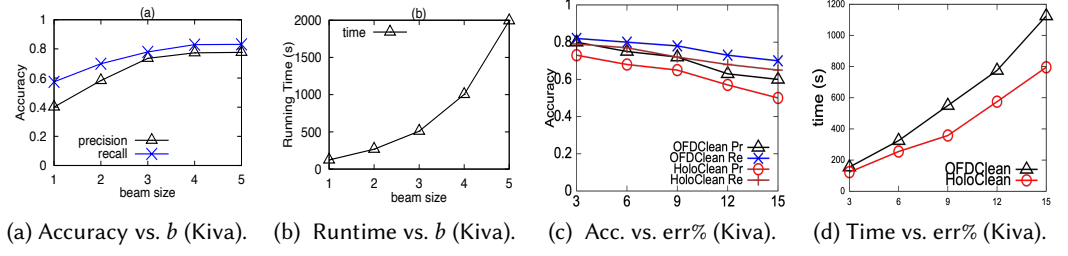
(a) Accuracy vs. *b* (Kiva). (b) Runtime vs. *b* (Kiva). (c) Acc. vs. err% (Kiva). (d) Time vs. err% (Kiva).

Fig. 12. OFDClean Performance.

8.5 OFDClean Performance

Exp-9: Vary beam size *b*. Figure 12a shows that as we evaluate more candidate ontology repairs, we achieve higher precision and recall. The incremental benefit becomes marginal once we have found the best repair, as reflected between $b = 4$ and $b = 5$. Figure 12b shows that the runtime increases exponentially due to the number of repair combinations that must be evaluated for increasing b. To manage runtime costs in practice, the beam size can be tuned according to accuracy requirements, with initial tuning guidelines given in Section 7.1.

Exp-10: Vary *err%*. For increasing error rates, Figure 12c shows that accuracy declines due to overlapping values between antecedent and consequent values among multiple OFDs. An update to attribute $A \in X_2$ for $\phi_1 : X_1 \rightarrow A$, $\phi_2 : X_2 \rightarrow B$, changes the distribution of equivalence classes w.r.t. X_2, leading to lower recall and precision. Figure 12d shows that runtimes increase as more errors are evaluated, and a larger space of repairs must be considered.

Exp-11: Vary *inc%*. Figure 10 shows the repair accuracy as we vary the incompleteness rate, *inc%*, which measures the percentage of values in I, but not in S, where such errors are resolved via ontology repairs. As *inc%* increases, the precision declines, as some repair values are added to the wrong sense. Recall scores are more consistent, achieving above 85%, with slight linear decline as error values are corrected by data updates.

Fig. 10. Acc. vs. *inc%*   Fig. 11. Acc. vs. $|\Sigma|$.

Exp-12: Vary $|\Sigma|$. Figure 11 shows that increasing the number of OFDs causes both precision and recall to decline as an increasing number of attributes overlap among the OFDs. OFDClean resolves errors when attribute overlap occurs in the consequent attribute between two OFDs, i.e., $\phi_1 : X_1 \rightarrow A_1$, $\phi_2 : X_2 \rightarrow A_2$, $A_1 = A_2$. However, when errors arise w.r.t. ϕ_1 due to an update in attribute $A_2 \in X_1$, we do not re-evaluate ϕ_1 again. We intend to explore this increased space of repairs in future work.

Exp-13: Vary *N*. Increasing the number of tuples, N, does not significantly impact the repair accuracy, with an average variation of 1.4% in precision (figure omitted for brevity). Similarly, OFDClean achieves linear runtime performance, as shown in Table 8. This runtime increase occurs due to the larger number of equivalance classes created by newly inserted domain values, and the overhead of evaluating ontology repair combinations.

Exp-14: Comparative Discussion. To the best of our knowledge, there are no ontology repair algorithms coupled with data repair, especially for sense selection. We compare OFDClean against HoloClean [40], which considers external information during repair selection. Figure 12c shows that OFDClean achieves increased precision and recall over HoloClean by 7.4% and 4.4%, respectively. By recognizing the same interpretation of different values via a sense, OFDClean selectively identifies
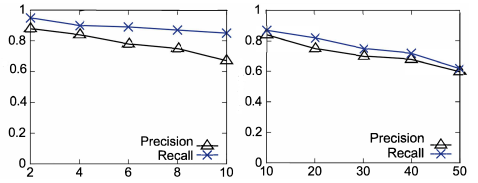
errors and improves precision. This comes at a cost in runtime (Figure 12d) since OFDClean must evaluate a space of ontology repairs to find those that minimize the number of data repairs.

9 RELATED WORK

Data Dependencies. Our work is similar to ontological dependencies and relaxed notions of FDs. Motik et al. define integrity constraints using the Web Ontology Language (OWL), where OWL ontologies are often incomplete [32]. They propose an extension of OWL to validate ontology completeness by defining inclusion dependencies and domain constraints to check for missing and valid domain values within an ontology. *Ontological Graph Keys (OGKs)* exploit ontologies to enhance keys for graphs [29]. An OGK incorporates an event pattern defined on an entity, which identifies instances w.r.t. concept similarity in an ontology *S*. OGKs characterize entity equivalence using ontological similarity. Our work shares a similar spirit to leverage ontological relationships and concepts. However, existing techniques do not consider the notion of senses to enable multiple interpretations of values in an ontology.

Past work has proposed relaxed notions of FDs including strong and weak FDs [25], and null FDs (NFDs) defined over incomplete relations (containing null values) [26]. A weak FD holds over a relation if there exists a possible world (instance) when an unknown value is updated to a non-null value. OFDs are similar to weak FDs when consequent values exist in the data but not in the ontology. We have shown that despite having equivalent axiom systems, the semantics of OFDs differ from NFDs, and verification of OFDs must be done over equivalence classes w.r.t. left-hand-side attributes of the OFD versus pairs of tuples for NFDs.

While OFD and NFD discovery can be modeled via a set-containment lattice, OFD discovery validates, for each equivalence class, whether there is a non-empty intersection of the senses. This incurs increased complexity during verification based on the number of synonyms and senses in the ontology. Computing data repairs to achieve consistency w.r.t. OFDs requires finding a sense that minimizes a cost function, e.g., the number of updates to achieve consistency, where interactions between OFDs, and finding the best sense, both incur additional challenges beyond NFDs.

Constraint-based Data Cleaning. FDs are used as a benchmark to propose data repairs such that the data and the FDs are consistent [8, 11, 38]. Extensions have relaxed the notion of equality in FDs, and used similarity and matching functions to identify errors, and propose repairs [6, 19]. While our work is in similar spirit, we differ in the following ways: (1) the similarity functions match values based only on syntactic string similarity; and (2) the notion of senses is not considered, and similarity under multiple interpretations is not possible.

Holistic Data Cleaning. Probabilistic approaches have shown promise to tackle the multiple facets of data cleaning. ERACER [31] and HoloClean [40] learn from given attribute templates and training samples to infer clean values. Existing systems either use a broad set of constraints [18], or bound repairs according to maximum likelihood [47]. These techniques capture context either via external sources or additional statistics. We advocate for a deeper integration of context into the data cleaning framework so that such "inconsistencies" are not flagged in the first place. To the best of our knowledge, the only prior work on using ontologies for data cleaning is KATARA [12], which includes simple patterns in ontologies such as "France" hasCapital "Paris", and does not integrate ontologies into the definition of integrity constraints. Context takes a central role in BARAN, which defines a set of error corrector models with a context-aware data representation to improve precision [30]. Incorporating these models into OFDClean and resolving conflicts between contextual models and ontologies are interesting avenues of future work.

Knowledge Base Incompleteness. Computational fact checking against Knowledge Bases (KBs) has emerged in recent years to automatically verify facts from different domains. The main problem in fact checking with KBs is that the reference information may be incomplete. That is, under an

Open World Assumption (OWA), we cannot assume that all facts in a KB are complete; facts not in a KB may be false or just missing [22]. In our work, we do not make such open assumptions. To improve KB quality, the RuDiK system discovers positive rules to mitigate incompleteness (e.g., if two persons have the same parent, then they are siblings), and negative rules to detect errors (e.g., if two persons are married, one cannot be a child of the other) [35]. Incorporating such inference into OFDClean as a new ontology repair operation will enrich existing ontological relationships.

10 CONCLUSIONS

Ontology Functional Dependencies (OFDs) capture domain relationships found in ontologies. We study fundamental problems for OFDs, and propose the FastOFD algorithm to discover a minimal set of OFDs. We also study the repair problem with respect to a set of OFDs, and propose an algorithm to find a sense for each equivalence class to provide an interpretation for each OFD against the data. Our proposed framework, OFDClean, combines sense selection with ontology and data repair such that the total number of updates to the data and to the ontology is minimized. Our experiments showed that our algorithms are effective, as OFDs are useful data quality rules to capture domain relationships, and significantly reduce the number of false positive errors in data cleaning solutions that rely on traditional FDs. As next steps, we intend to explore cleaning with respect to inheritance OFDs, which represent hierarchical relationships among data values.

REFERENCES

[1] [n.d.]. The drug ontology. http://www.ontobee.org/ontology/DRON.
[2] [n.d.]. The Kiva dataset. https://www.kaggle.com/kiva/data-science-for-good-kiva-crowdfunding?select=loan_themes_by_region.csv.
[3] 2016. Biomedical terminology, ontologies. *https://mor.nlm.nih.gov* (2016).
[4] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. 1996. Fast discovery of association rules. *Adv. in Knowl. Disc. & Data Mining* (1996), 307–328.
[5] S. Baskaran, A. Keller, F. Chiang, L. Golab, and J. Szlichta. 2017. Efficient discovery of ontology functional dependencies. In *CIKM*. 1847–1856.
[6] L. Bertossi, S. Kolahi, and L. Lakshmanan. 2011. Data Cleaning and Query Answering with Matching Dependencies and Matching Functions. In *ICDT*. 268–279.
[7] G. Beskales, Ihab F. Ilyas, Lukasz Golab, and Artur Galiullin. 2013. On the Relative Trust between Inconsistent Data and Inaccurate Constraints. In *ICDE*. 541–552.
[8] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. 2005. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*. 143–154.
[9] F. Chiang and R. J. Miller. 2011. A unified model for data and constraint repair. In *ICDE*. 446–457.
[10] X. Chu, I. Ilyas, and P. Papotti. 2013. Discovering Denial Constraints. *Proc. VLDB Endow.* 6, 13 (2013), 1498–1509.
[11] X. Chu, I. Ilyas, and P. Papotti. 2013. Holistic Data Cleaning: Putting Violations into Context. In *ICDE*. 458–469.
[12] X. Chu, J. Morcos, I. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. 2015. KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing. In *SIGMOD*. 1247–1261.
[13] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. 2007. Improving data quality: Consistency and accuracy. In *VLDB*. 315–326.
[14] Wenfei F., Jianzhong L., Shuai M., Nan T., and Wenyuan Y. 2011. Interaction between record matching and data repairing.. In *SIGMOD*. 469–480.
[15] Thomas S. Ferguson. 1989. Who Solved the Secretary Problem? *Statist. Sci.* 4, 3 (1989), 294–296.
[16] P. A. Flach and I. Savnik. 1999. Database Dependency Discovery: A Machine Learning Approach. *AI Commun.* 12, 3 (1999), 139–160.
[17] M. R. Garey and D. S. Johnson. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness.
[18] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. 2013. The LLUNATIC Data-Cleaning Framework. *PVLDB* 6, 9 (2013), 625–636.
[19] S. Hao, N. Tang, G. Li, J. He, N. Ta, and J. Feng. 2017. A Novel Cost-Based Model for Data Repairing. *TKDE* 29, 4 (2017), 727–742.
[20] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. 2009. *LinkedCT: A Linked Data Space for Clinical Trials.* Technical Report CSRG-596. University of Toronto.

[21] Y. Huhtala, J. Kinen, P. P., and H. Toivonen. 1998. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In *ICDE*. 392–401.

[22] V. Huynh and P. Papotti. 2018. Towards a Benchmark for Fact Checking with Knowledge Bases. In *Companion Proceedings of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 1595–1598.

[23] S. Kolahi and L. V. S. Lakshmanan. 2009. On approximating optimum repairs for functional dependency violations. In *ICDT*. 53–62.

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

[25] Mark Levene and George Loizou. 1998. Axiomatisation of Functional Dependencies in Incomplete Relations. *Theor. Comput. Sci.* 206, 1–2 (Oct. 1998), 283–300.

[26] Y. Edmund Lien. 1982. On the Equivalence of Database Models. *J. ACM* 29, 2 (1982), 333–362.

[27] S. Lopes, J. Petit, and L. Lakhal. 2000. Efficient Discovery of Functional Dependencies and Armstrong Relations. In *EDBT*. 350–364.

[28] B. Lowerre. 1976. The HARPY Speech Recognition System. *Ph.d. Thesis, Carnegie-Mellon University* (1976).

[29] H. Ma, M. Alipourlangouri, Y. Wu, F. Chiang, and J. Pi. 2019. Ontology-Based Entity Matching in Attributed Graphs. *Proc. VLDB Endow.* 12, 10 (2019), 1195–1207.

[30] M. Mahdavi and Z. Abedjan. 2020. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *Proc. VLDB Endow.* 13, 11 (2020), 1948–1961.

[31] C. Mayfield, J. Neville, and S. Prabhakar. 2010. ERACER: a database approach for statistical inference and data cleaning. In *SIGMOD*. 75–86.

[32] B. Motik, I. Horrocks, and U. Sattler. 2007. Adding Integrity Constraints to OWL. In *OWL: Experiences and Directions*.

[33] W. Ng. 2001. An Extension of the Relational Data Model to Incorporate Ordered Domains. *ACM TODS* 26, 3 (2001), 344–383.

[34] N. Novelli and R. Cicchetti. 2001. FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies. In *ICDT*. 189–203.

[35] S. Ortona, V. Meduri, and P. Papotti. 2018. Robust Discovery of Positive and Negative Rules in Knowledge Bases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 1168–1179.

[36] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, and F. Naumann. 2015. Data Profiling with Metanome. *PVLDB* 8, 12 (2015), 1860–1871.

[37] T. Papenbrock, S. Kruse, J. Quiané-Ruiz, and F. Naumann. 2015. Divide and Conquer-based Inclusion Dependency Discovery. *Proc. VLDB Endow.* 8, 7 (2015), 774–785.

[38] N. Prokoshyna, J. Szlichta, F. Chiang, R. Miller, and D. Srivastava. 2015. Combining Quantitative and Logical Data Cleaning. *PVLDB* 9, 4 (2015), 300–311.

[39] Yossi R., Carlo T., and Leonidas J. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision* 2, 40 (2000), 99–121.

[40] T. Rekatsinas, X. Chu, I. Ilyas, and C. Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10 (2017), 1190–1201.

[41] Peter J. Rousseeuw and Christophe Croux. 1993. Alternatives to the Median Absolute Deviation. *J. Amer. Statist. Assoc.* 88, 424 (1993), 1273–1283.

[42] Y. Rubner, C. Tomasi, and L. J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *Int. J. Comput. Vision* 40, 2 (2000), 99–121.

[43] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. 2017. Effective and Complete Discovery of Order Dependencies via Set-based Axiomatization. *Proc. VLDB Endow.* 10, 7 (2017), 721–732.

[44] J. Szlichta, P. Godfrey, and J. Gryz. 2012. Fundamentals of Order Dependencies. *PVLDB* 5, 11 (2012), 1220–1231.

[45] The Food and Drug Administration (FDA). 2018. National Drug Code Directory. https://www.fda.gov.

[46] C. Wyss, C. Giannella, and E. Robertson. 2001. FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining FDs from Relations. In *DaWaK*. 101–110.

[47] M. Yakout, L. Berti-Équille, and A. Elmagarmid. 2013. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *SIGMOD*. 553–564.

[48] H. Yao and H. Hamilton. 2008. Mining Functional Dependencies from Data. *Data Min. Knowl. Discov.* 16, 2 (2008), 197–219.

[49] Z. Zheng, L. Zheng, F. Chiang, L. Golab, and J Szlichta. 2021. Discovery and Contextual Data Cleaning with Ontology Functional Dependencies. *Technical report available at https://arxiv.org/abs/2105.08105, 28 pages* (2021).

[50] A. Ziawasch, P. Schulze, and F. Naumann. 2014. DFD: Efficient Functional Dependency Discovery. In *CIKM*. 949–958.