FastOFD: Contextual Data Cleaning with Ontology Functional Dependencies

Zheng Zheng
McMaster University
zhengz13@mcmaster.ca

Morteza Alipour Langouri
McMaster University
alipoum@mcmaster.ca

Zhi Qu, Ian Currie
McMaster University
{quz1,curriei}@mcmaster.ca

Fei Chiang
McMaster University
fchiang@mcmaster.ca

Lukasz Golab
University of Waterloo
lgolab@uwaterloo.ca

Jaroslaw Szlichta
University of Ontario IT
jaroslaw.szlichta@uoit.ca

ABSTRACT

Functional Dependencies (FDs) define attribute relationships based on syntactic equality. As a result, FD-based data cleaning systems incorrectly label syntactically different but semantically equivalent values as errors. To address this problem, we demonstrate FastOFD: a system for discovering Ontology Functional Dependencies (OFDs) which express semantic attribute relationships such as synonyms and is-a hierarchies defined by an ontology. In addition to discovering OFDs, FastOFD generates suggestions for repairing erroneous data, and we demonstrate that FastOFD significantly reduces the number of false positive errors compared to FD-based data cleaning techniques.

1 INTRODUCTION

In constraint-based data cleaning, tuples that violate the given integrity constraints are identified as erroneous. Candidate repairs are then generated to suggest how erroneous tuples could be modified to eliminate inconsistencies. The most popular integrity constraint considered in the data cleaning literature has been the Functional Dependency (FD) [2] and its extensions such as conditional FDs [3]. However, FDs are limited to identifying attribute relationships based on syntactic equivalence or syntactic similarity in case of metric FDs [5]. As a result, data cleaning systems that are based on FDs have a common flaw: they incorrectly label syntactically different but semantically equivalent values as errors. This leads to an increased number of reported "errors" and a larger search space of candidate data repairs.

Example: Table 1 shows a sample of clinical 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) violate F_1 as 'United States', 'America', and 'USA' are *not syntactically equivalent* (the same is true for (t_2, t_4, t_7)). However, 'United States' is synonymous with 'America' and 'USA', and (t_1, t_5, t_6) all refer to the same country. Similarly, 'Bharat' in t_4 is synonymous with 'India' as it is the country's original Sanskrit name. For F_2, (t_1, t_2, t_3) and (t_4, t_5, t_6) do not satisfy the dependency as the consequent values all refer to different medications. However, with domain knowledge from a medical ontology shown in Figure 2, we see that the values participate in an inheritance relationship. Both 'ibuprofen' and 'naproxen' are non-steroidal anti-inflammatory drugs (NSAID), and 'tylenol' is an 'acetaminophen' drug, which in turn is an 'analgesic'.

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

Table 1: Sample clinical trials data

To address these problems, we demonstrate FastOFD[1], a tool for *contextual* data cleaning with a novel class of dependencies called Ontology Functional Dependencies (OFDs) which take synonyms and inheritance relationships into account. In the above example, if F_1 and F_2 were specified as OFDs then no tuples would be falsely reported as erroneous. Our demonstration focuses on the following novel features of FastOFD:

(1) **Automatic discovery of OFDs to show how prevalent they are in real data.** We have recently proposed an efficient algorithm for discovering OFDs from data [1]. The FastOFD system uses this algorithm, and conference participants will be able to run it on several real datasets and ontologies, and visualize the results. Furthermore, conference participants will learn, through real examples, about an interesting aspect of OFDs that does not arise in standard FDs: the notion of senses or interpretations with respect to a given ontology. For example, the value "jaguar" can be interpreted as an animal or as a vehicle.

(2) **Data cleaning with OFDs.** FastOFD discovers OFDs that mostly hold but may be violated by a bounded number of tuples. Once such OFDs are discovered, FastOFD identifies erroneous tuples and suggests how to modify them in order to remove inconsistencies. Conference participants will be able to apply the suggested modifications (or propose different modifications) in real-time.

(3) **Comparison with FD-based data cleaning methods.** We demonstrate that FastOFD is practically as fast as existing algorithms for discovering traditional FDs, yet OFDs are more expressive. We also show that FastOFD significantly reduces the number of false positive errors compared to FD-based data cleaning techniques. This reduces the computational effort of data cleaning and the manual burden for users to identify falsely categorized data errors.

[1] http://db.cas.mcmaster.ca/fastofd

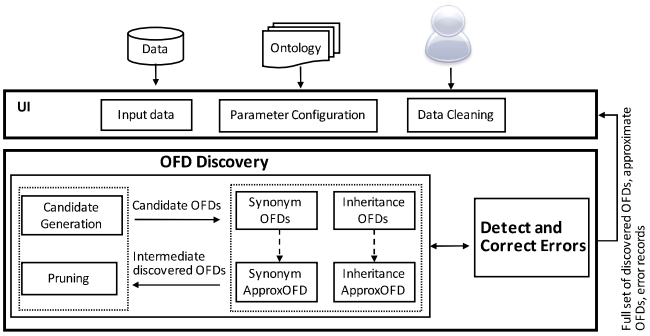

Figure 1: FastOFD Architecture.

2 SYSTEM OVERVIEW

2.1 Ontology 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. For example, in Table 1, $\Pi_{CC} = \{\{t_1, t_5, t_6\}\{t_2, t_4, t_7\}\{t_3\}\}$.

An ontology S is a specification of a domain that includes concepts, entities, properties, and relationships among them. The meaning of these constructs can be modeled according to different *senses* leading to different ontological interpretations. For example, the value 'jaguar' can be interpreted as an animal or as a vehicle. As an animal, 'jaguar' is synonymous with 'panthera onca', but not with the value 'jaguar land rover' which is an automotive company.

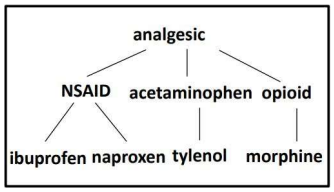

Figure 2: A medical ontology

We define classes E to capture the senses defined in S. Let $synonyms(E)$ be the set of all synonyms for a 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, $names(jaguar)$ = {$E2$, $E3$} as jaguar can be an animal or a vehicle. Let $descendants(E)$ be a set of all string representations for the class E or any of its descendants, i.e., $descendants(E) = \{s \mid s \in synonyms(E)$ or $s \in synonyms(E_i)$, where E_i *is-a* E_{i-1}, ..., E_1 *is-a* $E\}$. For instance, $descendants(E3)$ = {'jaguar', 'peruvian jaguar', 'mexican jaguar'}.

We consider two ontological relationships in the right-hand-side of a dependency, synonyms and inheritance, leading to the following definitions for synonym OFDs and inheritance OFDs:

Definition 2.1. A relation instance I satisfies a *synonym OFD* $X \rightarrow_{syn} A$, if for each equivalence class $x \in \Pi_X(I)$, there exists an interpretation in S under which all the A-values of tuples in x are synonyms. That is, $X \rightarrow_{syn} A$ holds if for each class x,

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

Definition 2.2. Let θ be a threshold representing the allowed path length between two nodes in S over an attribute A (each attribute can have a different θ). A relation instance I satisfies an *inheritance OFD* $X \rightarrow_{inh} A$ if for each equivalence class $x \in \Pi_X(I)$, the A-values of all tuples in x are descendants of a Least Common Ancestor (LCA) which is within a distance of θ in S. That is, $X \rightarrow_{inh} A$ holds if for each equivalence class x,

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

and the resulting LCA is within a distance of θ in S to each a.

Example: Consider the OFD ϕ_1: [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') ∩ names('America') ∩ names('USA') = 'United States of America'. Similarly, the second class $\{t_2, t_4, t_7\}$ gives names('India') ∩ 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, ϕ_1 holds over Table 1. Now consider the OFD ϕ_2: [SYMP, DIAG] $\rightarrow_{inh}$ [MED], and the ontology in Figure 2. For the first equivalence class, $\{t_1, t_2, t_3\}$, the LCA is 'NSAID' which is within a distance of one to each MED value in this class. For the second equivalence class, $\{t_4, t_5, t_6\}$, the LCA is 'analgesic', which is within a distance of two to each MED value in this class. The third equivalence class consists of a single tuple (t_7) so there is no conflict. Thus, ϕ_2 holds with $\theta = 2$ over MED.

Synonym OFDs subsume traditional FDs, where all values are assumed to have a single canonical name (i.e., for all classes E, $|synonyms(E)| = 1$). Furthermore, inheritance OFDs subsume synonym OFDs since we can recover synonym OFDs by setting $\theta = 0$ for each attribute. For example, the inheritance OFD [SYMP, DIAG] $\rightarrow_{inh}$ [MED] allows synonyms such as 'ibuprofen' and 'advil' to appear in tuples having the same SYMP and DIAG values; however, different medications under the same LCA such as 'ibuprofen' and 'naproxen' are not allowed.

OFDs *cannot* be reduced to traditional FDs or Metric FDs [5] (which assert that two tuples whose left-hand side attribute values are equal must have syntactically similar righthand side attribute values according to some distance metric). Since values may have multiple senses (e.g., jaguar the animal and jaguar the car), 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).

2.2 OFD Discovery

The notion of senses makes OFDs non-trivial and has important implications on their discovery. While checking pairs of tuples is sufficient to identify violations and therefore verify traditional FDs (or metric FDs), this is not the case for OFDs. To verify an OFD, we must find a common interpretation of the Y-values for each equivalence class, as shown in Definitions 2.1 and 2.2. As a result, existing dependency discovery algorithms that validate dependencies via pairwise tuple comparisons (e.g., FastFD [6]) cannot be easily extended to OFDs.

Instead, we use our OFD discovery algorithm [1] which uses an Apriori-like approach, similar to the TANE FD discovery algorithm [4]. In this approach, the set of possible antecedent and consequent values considered by FastOFD is modeled as a set containment lattice. 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 to multi-attribute sets until (potentially) level $k = n$, where n is the number of attributes. 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 synonym or inheritance OFD as per Definition 2.1 and Definition 2.2. This small-to-large search strategy guarantees that only minimal OFDs are discovered, and we develop novel optimizations to prune the search space effectively [1]. In our evaluation, we found that FastOFD incurred an average overhead of 5% to 10% over TANE [4] while discovering a larger set of (synonym, inheritance and approximate) dependencies.

2.3 Data Cleaning with Approximate OFDs

In addition to OFDs that hold over the entire data instance, FastOFD can discover approximate OFDs that hold with some exceptions. We define a minimum support level, τ, that specifies the minimum number of tuples that must satisfy an OFD ϕ. Given a relational instance I, and a minimum support threshold $\tau, 0 \leq \tau \leq 1$, FastOFD can find all minimal OFDs ϕ such that $s(\phi) \geq \tau$ where $s(\phi) = max\{|r| \mid r \subseteq I, r \models \phi\}$. Since approximate OFDs are satisfied by most of the relation, violating tuples may be considered erroneous. These are records involving syntactically different but semantically equivalent attribute values that would otherwise be identified as errors using traditional FDs.

Furthermore, unlike previous dependency discovery algorithms, FastOFD discovers OFDs that hold under a given sense, which provides context for data cleaning. For a given ontology, we assume a sense is provided that defines the ontological context under which we identify the exceptions and the corresponding fixes (based on satisfying record values). For synonym OFDs, the set of potential clean values includes all synonyms defined in the ontology. For inheritance OFDs, we identify the highest ancestor a of the frequent (clean) values in the data, and record the set of ontology values from the sub-tree with a as the root. We ensure that the height of the sub-tree does not exceed θ.

2.4 FastOFD Architecture

FastOFD is an interactive web application running on Flask + uWSGI using Python and JavaScript libraries. The backend is implemented using Java v1.8, running on an Intel Xeon CPU E7-LL8867 2.13GHz with 32GB of memory. Figure 1 shows the FastOFD architecture, consisting of a user-interface (UI) layer, the OFD discovery engine, and a data cleaning module that identifies and corrects error values by interactively engaging with a user. The user interface (UI) layer provides the input specifications for the data instance, parameter configuration, and ontology RDF files along with the corresponding attributes and senses. Configuration settings include specifying the path length threshold (θ), the desired type of OFDs to be discovered (synonym or inheritance, or both), whether approximate OFDs are desired, and if so, the minimum support level. Discovered OFDs are returned to the user along with approximate OFDs for interactive cleaning.

The discovery algorithm generates candidate OFDs by traversing the attribute lattice in a level-wise manner. FastOFD checks whether a candidate $\phi : X \rightarrow A$ satisfies a synonym or inheritance OFD relative to the given parameter settings. If so, then candidate OFDs that are supersets of X are pruned from the lattice, and ϕ is added to the list of discovered OFDs. If ϕ is an approximate OFD (i.e., with support greater than τ), we record the satisfying (clean) values along with the exceptions, and continue the search by evaluating candidate OFDs containing supersets of X. The efficiency of our algorithm relies on pruning candidates

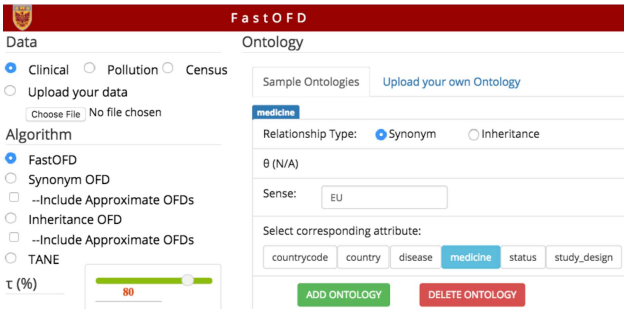

Figure 3: Input and parameter configurations.

through our axioms from the lattice that are supersets of discovered OFDs, keys, and FDs. We also disqualify approximate OFD candidates that do not have a minimum τ support. That is, for a candidate $X \rightarrow A$, adding a new attribute B to X only fragments the equivalence classes, and does not increase the support level.

3 DEMONSTRATION OVERVIEW

In this demo, conference participants will interact with FastOFD to: (1) discover OFDs and see first-hand how prevalent they are in real data. We show examples from both synonym and inheritance OFDs, and cases where similar OFDs exist under different senses; (2) interactively clean the data using approximate OFDs where candidate repairs are provided, and users can apply these changes in real-time with immediate feedback on the impact to support levels; and (3) experience the effectiveness of FastOFD to reduce the number of false positive errors while achieving comparable runtimes to traditional FD discovery algorithms.

Input Interface. Figure 3 shows the input interface where users can select a sample dataset among clinical trials data from LinkedCT.org, pollution data from the Canadian Pollutant Release Inventory, and census-income data from https://archive.ics.uci.edu/ml/. In this paper, we use the clinical data to demonstrate example scenarios. Users can then select the type of dependencies, configure τ to return approximate OFDs with minimum τ support, and inheritance OFDs containing IS-A relationships within a distance of θ in the ontology. For each input ontology, users can define the corresponding synonym or inheritance relationship, the applicable attribute, and the sense interpretation. For example, in Figure 3, we add an ontology containing synonyms for attribute 'medicine' to be interpreted under the sense 'EU' for European Union.

3.1 Automatic Discovery of OFDs

Figure 4 shows an example output of two discovered synonym OFDs, and four approximate synonym OFDs ranked according to decreasing support. Each dependency shows the corresponding sense under which it holds over the data. The inclusion of senses is unique to FastOFD (in contrast to other dependency discovery systems) as it provides a specific interpretation under which an OFD holds. For example, Figure 4 shows that the synonym OFD [disease] → [medicine] holds over the clinical data under two senses: (i) as an OFD under sense 'EU' (for the 'European Union'), and (ii) as an approximate (synonym) OFD under sense 'US' (for the 'United States') with 93.4% support. This example shows that the same medication may be used to treat different diseases, and is referred to by different names, in different regions of the world (i.e., in different geographical senses). Senses help to differentiate semantically equivalent entities under different contexts. Similarly, the OFD [country] → [countryCode] holds

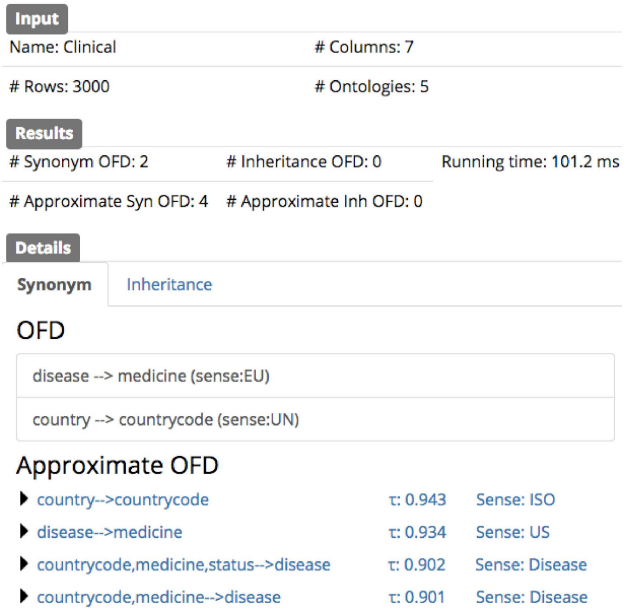

Figure 4: Viewing discovered OFDs.

under the sense 'UN' denoting country name abbreviations published by the United Nations, but holds approximately under the sense 'ISO' denoting the abbreviations used by the International Organization for Standardization. Conference participants will be able to delve deeper (by clicking on each OFD) to explore these distinctions between similar dependencies. Unlike previous dependency discovery algorithms that assume equivalence classes, FastOFD provides a contextual view for discovered OFDs that can be customized with interpretations according to a user's domain or application requirements.

3.2 Contextual Data Cleaning

Given a set of approximate OFDs, we now show how users can interact with FastOFD to correct these exceptions. Figure 5 shows a data cleaning example w.r.t. the OFD ϕ: [disease] $\rightarrow$ [medicine], that states a given disease is treated by prescribed medications. However, while ϕ holds over the clinical data under the sense 'EU', it holds only approximately under the 'US' sense as medication names vary between the two regions. An example of this semantic data quality issue is shown in Figure 5 by expanding the approximate OFD version of ϕ. Users can view the positive (supporting) examples highlighted in green, while the exceptions are highlighted in red. In this example, 'asthma' is treated by medications {'Advicor', 'Advair', 'Seretide'}, which are all synonymous in the 'EU' sense. However, in the 'US', 'Seretide' is not recognized as a synonym, and is flagged as an exception. The frequency levels for each tuple pattern are shown such that users can correct exceptions directly, and receive immediate feedback. By clicking 'Apply', if the entered value(s) are correct, the corresponding τ support level for the OFD is updated, and the tuple pattern changes from red to green. Conference participants will be able to perform hands-on exploration and cleaning, and observe how prevalent these OFDs are in real data.

3.3 Effectiveness of FastOFD

Finally, we show that FastOFD significantly reduces the number of 'false positive' errors that are found in FD-based data cleaning solutions, with comparable running times to FD discovery algorithms even though OFDs have greater expressiveness and

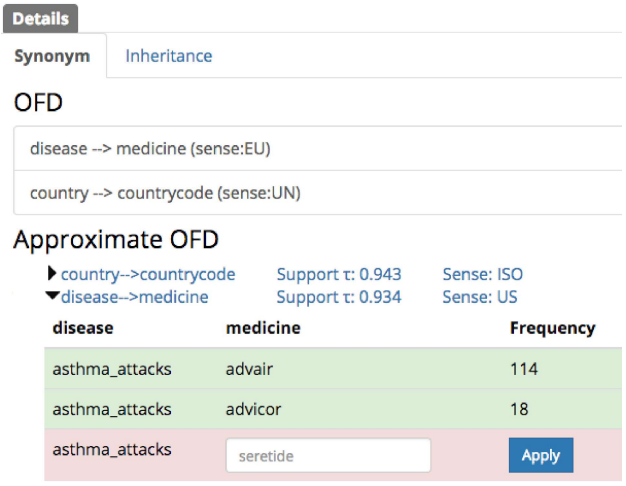

Figure 5: Data cleaning with approximate OFDs.

larger result sets [1]. Figure 6 shows the comparative performance between FastOFD and TANE, and more importantly, the distinctions between the two sets of approximate dependencies. While the same set of dependencies are discovered, the support level τ is higher for approximate OFDs than approximate FDs. Upon closer inspection, users will be able to drill down and notice that these differences are caused not only due to true errors (highlighted in red), but also the false positive errors (highlighted in blue). These false positive values represent values that are *syntactically* different than the clean (green highlighted) values, but are *semantically* equivalent. For example, the medication 'Celexa' is synonymous with 'Celebrex' in treating 'osteoarthritis'. These values are identified as clean in FastOFD but as exceptions in TANE. Our case studies show that an average 33% of errors found in traditional FD-based data cleaning solutions are false positive errors that can be eliminated. This reduces the manual burden of having users validate these 'errors', and also reduces the space of possible repairs, thereby improving the overall data cleaning runtime. In addition to the synonym examples described here, conference participants will be able to explore examples involving inheritance properties that are present in real data.

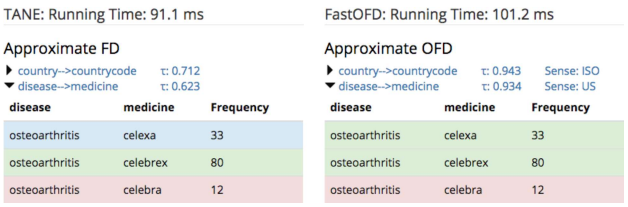

Figure 6: Reducing the number of false positives.

REFERENCES

[1] S. Baskaran, A. Keller, F. Chiang, L. Golab, and J. Szlichta. Efficient discovery of ontology functional dependencies. In *CIKM*, pages 1847–1856, 2017.
[2] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, pages 143–154, 2005.
[3] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *VLDB*, pages 315–326, 2007.
[4] Y. Huhtala, P. P. J. Kinen, and H. Toivonen. Efficient discovery of functional and approximate dependencies using partitions. *ICDE*, pages 392–401, 1998.
[5] N. Prokoshyna, J. Szlichta, F. Chiang, R. Miller, and D. Srivastava. Combining quantitative and logical data cleaning. *PVLDB*, 9(4):300–311, 2015.
[6] C. Wyss, C. Giannella, and E. L. Robertson. FastFDs: Heuristic-driven, depth-first alg. for mining FDs from relations. In *DaWaK*, pages 101–110, 2001.