On Axiomatization and Inference Complexity over a Hierarchy of Functional Dependencies

Jaroslaw Szlichta[1], Lukasz Golab[2], and Divesh Srivastava[3]

[1] University of Ontario Institute of Technology, Oshawa, Canada
jaroslaw.szlichta@uoit.ca
[2] University of Waterloo, Waterloo, Canada
lgolab@uwaterloo.ca
[3] AT&T Labs-Research, New Jersey, USA
divesh@research.att.com

Abstract. Functional dependencies (FDs) have recently been extended for data quality purposes with various notions of similarity instead of strict equality. We study these extensions in this paper. We begin by constructing a hierarchy of dependencies, showing which dependencies generalize others. We then focus on an extension of FDs that we call Antecedent Metric Functional Dependencies (AMFDs). An AMFD asserts that if two tuples have *similar* but not necessarily equal values of the antecedent attributes, then their consequent values must be equal. We present a sound and complete axiomatization as well as an inference algorithm for AMFDs. We compare the axiomatization of AMFDs to those of the other dependencies, and we show that while the complexity of inference for some FD extensions is quadratic or even co-NP complete, the inference problem for AMFDs remains linear, as in traditional FDs.

1 Introduction

Poor data quality is a bottleneck to effective business decision-making. Big data initiatives are likely to take longer, cost more, and deliver fewer benefits without clean data. The ability to store data is no longer a problem: according to a survey of 586 senior executives conducted in June 2011 by the Economist Intelligence Unit (EIU) [1], less than 20% indicated data storage as a problem; however more than 50% rated data management tasks such as cleaning as problematic.

With the interest in data analytics at an all-time high, data quality has become a critical issue in research and practice. Integrity constraints, which specify the intended semantics and attribute relationships, are commonly used to characterize and ensure data quality [6, 20]. In particular, Functional Dependencies (FDs), which have traditionally been used in schema design, have recently been extended for data consistency purposes. An FD asserts that if two tuples agree on the left-hand-side attributes, then they must also agree on the right-hand-side attributes. The idea behind various extensions of FDs is to replace strict equality with some notion of *similarity*, either on the left-hand-side (see, e.g,

Table 1: Movie Relation

source	title	length	year	director
A	A Beautiful Mind	135	2001	Ridley Scott
B	A Beaut. Mind	135	2001	Ridley Scott
C	Beautiful Mind	135	2001	Ridley Scott

Matching Dependencies [7, 5, 9]), on the right-hand-side (see, e.g, Metric Functional Dependencies [13] and Sequential Dependencies [11]), or on both sides of the dependency (see, e.g., Differential Dependencies [16]).

In this paper, we study these generalizations of FDs. Our first objective is to construct a hierarchy of dependencies, revealing which ones (strictly) generalize others, and comparing their axiomatization and complexity of inference.

We then introduce a particular generalization of FDs that we call Antecedent Metric FDs (AMFDs). An AMFD asserts that if two tuples have *similar* but not necessarily equal values of the antecedent attributes, then their consequent values must be equal; we will compare AMFDs with related dependencies in Section 2.

To illustrate the utility of AMFDs, consider the movie data set shown in Table 1, which was put together from multiple data sources. In the process of merging data from various sources, it is often the case that small variations occur. For example, one source might report the movie *A Beautiful Mind* to have a running time of *135* minutes, as shown in Table 1, while another source may refer to the same movie as *A Beaut. Mind* and the third one as *Beautiful Mind*. An AMFD $\{title,\ year,\ director\} \mapsto \{length\}$ indicates that movies with similar titles, years, and directors (up to some distance threshold, as we will discuss in Section 2) must have equal lengths. Of course, we assume that the semantics are such that two similar movie titles, made in similar years, by similar director names do in fact refer to the same movie.

An FD $\{title,\ year,\ director\} \rightarrow \{length\}$ would not require the three *length* values in Table 1 to be equal, even though they refer to the same movie. Thus, AMFDs generalize FDs and can express the additional semantics of similarity.

The inference problem is to determine whether a dependency is logically entailed by a set of dependencies. For FDs, the inference problem has been well studied in previous work [3, 4]. We prove that while AMFDs are more expressive than FDs and have a more complex axiomatization, their complexity of inference remains linear.

The contributions of this paper are as follows.

1. *Hierarchy:* we construct a hierarchy of dependencies, showing which ones generalize others and comparing their complexity of reasoning. Our hierarchy shows which dependencies are practical and which are hard to reason about, and suggests further research on identifying tractable extensions of FDs.
2. *FD extension:* we introduce AMFDs, which describe integrity constraints on tuples with similar attribute values and are useful in data cleaning.
3. *Axiomatization:* we present a sound and complete axiomatization for AMFDs. Axiomatization is a first necessary step to designing an efficient inference procedure. Our axiomatization reveals interesting insights about inference

Table 2: Notational Conventions

Relations
- A bold capital letter represents a relation schema: **R**. Italic capital letters near the beginning of the alphabet represent single attributes: A and B.
- A small bold capital letter in italic represents a relation (a table): $\mathbf{t}$.
- Small italic letters near the end of the alphabet denote tuples: s and t.
- Small italic letters near the beginning of the alphabet denote attribute values: a, b and c. A small italic letter m denotes a similarity metric.

Sets
- Italic capital letters near the end of alphabet stand for sets of attributes: X
- XY is shorthand for $X \cup Y$. Likewise, AX or XA stand for $X \cup \{A\}$.

rules over AMFDs. For instance, the Reflexivity and Augmentation axioms, which hold for traditional FDs, are not necessary true for AMFDs.

4. *Inference Procedure:* we develop an inference procedure for AMFDs that runs in linear time in the complexity of the schema[4]. We implemented the inference algorithm and experimentally verified its efficiency.

The remainder of this paper is organized as follows. In Section 2, we review previous work, formally define AMFDs, and present a hierarchy of dependencies. In Sections 3 and 4, we present a sound and complete axiomatization and an inference procedure for AMFDs, respectively, and we compare the axiomatization to those of other related dependencies. We conclude the paper in Section 5.

2 Fundamentals

2.1 AMFDs

We provide notational conventions in Table 2. To accommodate small variations in the attribute values on the left-hand-side of the dependency, we define AMFDs (Definition 2). This is a departure from traditional FDs which enforce equality on both sides. Before we define AMFDs, we first define a *similarity* operator with a distance threshold.

Definition 1. *(similarity) For every attribute A in a relational schema $\boldsymbol{R}$, we assume a binary similarity relation ($\approx_{m,\theta}$) w.r.t. some similarity metric m and a threshold parameter $\theta \geq 0$. Specifically, for two tuples s and t, $s[A] \approx_{m,\theta} t[A]$ iff $m(s[A], t[A]) \leq \theta$. Metric m satisfies standard properties; it is symmetric, satisfies the triangle inequality and identity of indiscernibles, i.e., $m(a, b) = 0$ iff $a = b$. For two tuples s, t in relation $\boldsymbol{t}$ over $\boldsymbol{R}$, we write $s[X] \approx_{\boldsymbol{m},\Theta} t[X]$ to mean $s[A_1] \approx_{m_1,\theta_1} t[A_1]$, ..., $s[A_n] \approx_{m_n,\theta n} t[A_n]$, where $X = \{A_1, ..., A_n\}$, $\boldsymbol{m} = [m_1, ..., m_n]$ and $\Theta = [\theta_1, ..., \theta_n]$.*

[4] Our inference procedure is efficient because it is done at the schema level, which is much smaller than the size of the data.

Next, we define AMFDs. By definition, AMFDs generalize FDs.

Definition 2. *(AMFD) Let X and Y be two sets of attributes, and let $\boldsymbol{m}_X$ and Θ_X be metrics and thresholds for attributes X. Then, $X \mapsto Y$ denotes an antecedent metric FD (AMFD), read as X metrically functionally determines Y. Let $\boldsymbol{R}$ be a relation schema that contains the attributes that appear in X and Y, and let $\boldsymbol{t}$ be a relation instance of $\boldsymbol{R}$. Relation $\boldsymbol{t}$ satisfies $X \mapsto Y$ ($\boldsymbol{t} \models X \mapsto Y$), iff for all tuples s, $t \in \boldsymbol{t}$, $s[X] \approx_{\mathbf{m}_X, \Theta_X} t[X]$ implies $s[Y] = t[Y]$. An AMFD $X \mapsto Y$ is said to hold for $\boldsymbol{R}$, written as $\boldsymbol{R} \models X \mapsto Y$, iff for each admissible relational instance $\boldsymbol{t}$ of $\boldsymbol{R}$, relation $\boldsymbol{t}$ satisfies $X \mapsto Y$. An AMFD $X \mapsto Y$ is trivial iff for all $\boldsymbol{t}$, $\boldsymbol{t} \models X \mapsto Y$.*

Example 1. (AMFD) Assume metrics m_{title} and $m_{director}$ are edit distances with thresholds $\theta_{title} = 6$ and $\theta_{director} = 0$, respectively, in Table 1 (movie relation). Also assume that metric m_{year} is an integer distance with a threshold $\theta_{year} = 0$. Therefore, Table 1 satisfies the AMFD $\{title,\ year,\ director\} \mapsto \{length\}$.

2.2 Related Work

AMFDs and traditional FDs are specified over a *single* relation. However, AMFDs replace strict equality on the *left-hand-side* of the dependency with similarity. Dependencies defined over a single relation with similarity on the *right-hand-side*, called Metric FDs, were proposed by Koudas et al. [13]. We call them Consequent Metric FDs (CMFDs) to distinguish them from AMFDs. The verification problem over CMFDs was studied in [13], which is to decide whether the instance satisfies a prescribed set of dependencies. However, axiomatization and inference were not considered.[5]

Bertossi et al. [5], Fan [7] and Fan et al. [9] studied Matching Dependencies (MDs), which are object-identification constraints across *multiple* relations. MDs also enforce similarity rather than equality on the left-hand-side, but allow arbitrary Boolean similarity functions. (These similarity functions only need to satisfy reflexivity, symmetry and subsumption of equality.) On the other hand, AMFDs are defined over a single relation and only allow a restricted notion of similarity, namely thresholds over similarity metrics (recall Definition 1). Fan et al. presented a sound and complete axiomatization[6] and a quadratic-time inference procedure for MDs.

Pointwise Order Dependencies (PODs) [10] consider order relationship rather than equality of attribute values. A relation satisfies a POD $X \hookrightarrow Y$ if, for all tuples s and t, for every attribute A in X, $s_A\ op\ t_A$ implies that for every attribute B in Y $s_B\ op\ t_B$, where $op\ \epsilon\{<, >, \leq, \geq, =\}$. For example, in relation

[5] Some of the authors of this paper solved the axiomatization and inference problems for CMFDs in a paper currently under submission.

[6] It is stated in Fan et al. [9] (without a proof of completeness) that a complete axiomatization for MDs consists of 11 axioms, but only 9 sound axioms are presented.

Table 3: TimePolls Relation

sequential_id	timestamp	date	year	month	day
1	20140201142320	20140201	2014	02	01
2	20140201142325	20140202	2014	02	02

TimePolls (Table 3), the POD $\{date^{>}\} \hookrightarrow \{year^{=}, month^{=}, day^{>}\}$ holds; however, the POD $\{date^{>}\} \hookrightarrow \{year^{=}, month^{=}, day^{\leq}\}$ does not hold. Ginsburg and Hull [10] present a sound and complete axiomatization for PODs and show that the inference problem for them is co-NP-complete.

PODs are defined over sets of attributes. On the other hand, Lexicographical Order Dependencies (LODs) are defined over lists of attributes [17, 19]. LODs describe the relationship among lexicographical orderings of sets of tuples. This is the notion of order used in SQL and in query optimization, as per the *order by* operator (nested sort). A relation satisfies a LOD $\mathbf{X} \hookrightarrow \mathbf{Y}$ if any list of its tuples that satisfy *order by* $\mathbf{X}$ also satisfies *order by* $\mathbf{Y}$; however, not necessarily vice versa. ($\mathbf{X}$ and $\mathbf{Y}$ denote lists of attributes.) For instance, in relation TimePolls, the LODs *timestamp* $\hookrightarrow$ *date* and [*date*] $\hookrightarrow$ [*year, month, day*] are true. The default direction of the SQL order by is ascending. This can be generalized to order-by's that mix *asc* and *desc* directions, e.g., *order by name asc, age desc*. For example, in relation TimePolls, the LOD [$-$*sequential_id desc*] $\hookrightarrow$ [*timestamp asc*] holds. Szlichta et al. present a sound and complete axiomatization for lexicographical order dependencies and show that the inference problem for LODs is co-NP-complete [17, 19].

Another constraint for ordered data, sequential dependencies (SDs), was introduced in Golab et al. [11]. For example, the SD *sequential_id* $\hookrightarrow_{[4,5]}$ *timestamp* means that after sorting the data by the attribute *sequential_id*, the *gaps* between consecutive timestamps are between 4 and 5. This particular SD holds in the *TimePolls* relation; however, the SD *sequential_id* $\hookrightarrow_{[6,7]}$ *timestamp* does not hold. Golab et al. present a framework for discovering which subsets of the data obey a given SD, but axiomatization and inference were not considered.

SDs were generalized in Song and Chen [16] by introducing gaps (differential functions) on both sides of the dependency and named Differential Dependencies (DDs). For instance, in the relation TimePolls, the DDs $sequential\_id^{[1,1]} \hookrightarrow timestamp^{[4,5]}$ and $\{date^{[0,1]}\} \hookrightarrow \{year^{[0,0]}, month^{[0,0]}, day^{[0,1]}\}$ hold. However, the DD $sequential\_id^{[1,1]} \hookrightarrow timestamp^{(5,6]}$ does not hold. Song and Chen present an axiomatization and show that inference problem for DDs is co-NP-complete.

2.3 Hierarchy of Dependencies

Figure 1 illustrates a hierarchy of the dependencies we discussed above as well as a new class: MDDs. MDDs strictly generalize MDs and DDs by allowing differential functions (on the left hand side and the right hand side) with arbitrary similarity functions and allowing multiple tables. Below each dependency name, we point out the complexity of inference. Observe that for the "not studied"

dependencies, their complexity of inference is bookended by their immediate ancestors and descendants in the hierarchy.

We say that a dependency class $\mathcal{A}$ *generalizes* dependency class $\mathcal{B}$ *iff* there is a semantically preserving mapping of any dependency of class $\mathcal{B}$ into a set of dependencies of class $\mathcal{A}$. Class $\mathcal{A}$ *strictly generalizes* class $\mathcal{B}$ *iff* $\mathcal{A}$ generalizes $\mathcal{B}$, however, $\mathcal{B}$ does not generalize $\mathcal{A}$. The hierarchy in Figure 1 shows which dependencies strictly generalize others; due to space constraints, proofs will appear in extended version of this paper.

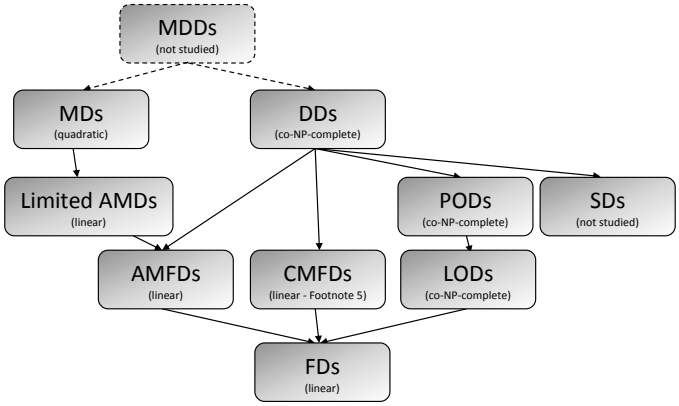

Fig. 1: Hierarchy of dependencies and their complexity of inference.

For example, DDs strictly generalize SDs. In our example involving Table 3, with the SD $sequential\_id \hookrightarrow_{[4,5]} time$, consecutive sequence numbers can be simulated by using on the left-hand-side of the DD a similarity metric which returns distance *one* if two numbers are consecutive and *zero* otherwise. Axiomatization and complexity of inference for SDs are open problems. However, since our hierarchy indicates that DDs strictly generalize SDs, the upper bound on the complexity of inference for SDs is co-NP complete.

Similarly, DDs strictly generalize PODs (which strictly generalize LODs [19]). For instance, a POD $A^{\geq} \hookrightarrow B^{\leq}$ is equivalent to a DD $A^{[0;+\infty]} \hookrightarrow B^{[-\infty;0]}$. This suggests that the complexity results for PODs can be adapted to DDs and did not have to be re-developed from scratch in [16]. AMFDs and CMFDs are also subsumed by DDs, since DDs allow similarity both on the left-hand-side and right-hand side. (Limited AMDs are introduced in Section 3.) Both AMFDs and CMFDs strictly generalize FDs by replacing equality with similarity.

3 Axiomatization

3.1 Soundness and Completeness

We now present an axiomatization for AMFDs, analogous to Armstrong's axiomatization for FDs [3, 4]. This provides a formal framework for reasoning about

1. *Void*
   $X \mapsto \{\}$
2. *Transitivity*
   If $X \mapsto Y$
   and $Y \mapsto Z$
   then $X \mapsto Z$
3. *Composition*
   If $X \mapsto Y$ and $Z \mapsto W$
   then $XZ \mapsto YW$
4. *Decomposition*
   If $X \mapsto Y$ and $Z \subseteq Y$
   then $X \mapsto Z$
5. *Reduce*
   If $XZ \mapsto Y$ and $X \mapsto Z$
   then $X \mapsto Y$
6. *Limited Reflexivity*
   If $Y \subseteq X$ and $\Theta_Y = 0$
   then $X \mapsto Y$

Fig. 2: Axiomatization for AMFDs.

AMFDs. The axioms give insights into how AMFDs behave and reveal how dependencies logically follow from others, which is not easily evident when reasoning from first principles. Also, a sound and complete axiomatization is necessary for an efficient inference procedure (see Section 4).

The axioms for AMFDs are presented in Figure 2. Recall that $\{\}$ denotes an empty set. Two of the axioms generate trivial dependencies that are always true: Void and Limited Reflexivity. Below we introduce additional inference rules that follow from the axioms in Figure 2. These will be used throughout the paper, particularly to prove that our AMFD axioms are complete.

Lemma 1. *(Left Augmentation) If $X \mapsto Y$, then $XZ \mapsto Y$.*

Proof. By Void and Composition it follows that $XZ \mapsto Y$. □

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

Proof. By Composition it follows that $X \mapsto YZ$. □

Next, we define *closure* over AMFDs. The closure of a set of attributes X is the set of attributes that X logically determines given a set of AMFDs F.

Definition 3. *(Closure X^+) The AMFD-closure of set of attributes X, denoted X^+, w.r.t. a set of AMFDs F using axioms $I = \{1\text{–}6\}$ in Figure 2, is defined as, $X^+ = \{A \mid F \vdash X \mapsto A\}$.*

Lemma 3 tells us whether a dependency follows from F using our axioms.

Lemma 3. *(Closure for AMFDs) $F \vdash X \mapsto Y$, if and only if $Y \subseteq X^+$.*

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

Theorem 1. *(Completeness) AMFD axioms are sound and complete.*

Proof. The soundness proof (if $F \vdash X \mapsto Y$, then $F \models X \mapsto Y$) is trivial. We just have to show that each axiom is true. We present the completeness proof (if $F \models X \mapsto Y$, then $F \vdash X \mapsto Y$). We consider a table $\mathbf{t}$ with two rows, whose template is shown in Table 4. We divide the attributes of $\mathbf{t}$ into three subsets: X_+, the set N, consisting of attributes in X that are not in the closure of X[7],

[7] For a traditional FD $X \mapsto Y$, by Reflexivity all the attributes in X are also in X^+. However, this is not true for AMFDs, as we will show in Example 2.

Table 4: Table template for AMFDs.

X^+	$N = \{A \mid A \in X \text{ and } A \notin X^+\}$	other attributes
$a...a$	$a...a$	$a...a$
$a...a$	$b...b$	$c...c$

and all the remaining attributes. All the attributes of the first row have the value a, while for the second row, the attributes in X^+ are a's, the attributes in N are b's and the other attributes are c's. Without loss of generality, assume that for all the attributes A in N and *other attributes* over $\mathbf{t}$ we use the same metric m, and that a and b are similar ($a \approx_{m,\theta_A} b$) but not equal. Also, assume that the values a and c are not similar ($a \not\approx_{m,\theta_A} c$), and hence not equal.

We first show that all dependencies in the set of AMFDs F are satisfied by table $\mathbf{t}$ ($\mathbf{t} \models F$). Since the AMFD axioms are sound, AMFDs inferred from F are true. Note that by Void and Limited Reflexivity, all trivial AMFDs are satisfied in table $\mathbf{t}$. Assume $V \mapsto Z$ is in F but is not satisfied by table $\mathbf{t}$. Therefore, $V \subseteq \{X^+ \cup N\}$ because otherwise two rows of $\mathbf{t}$ are not similar on some attribute of V since $a \not\approx_{m,\theta_A} c$, and consequently an AMFD $V \mapsto Z$ would not be violated. Moreover, Z cannot be a subset of X^+ ($Z \not\subseteq X^+$), or else $V \mapsto Z$ would be satisfied by $\mathbf{t}$. Let A be an attribute of Z not in X^+. Since the dependency $V \mapsto Z$ is in F, by Decomposition, $V \mapsto A$. Let V_1 be a maximal set of attributes such that $V_1 \subseteq V$ and $V_1 \subseteq X^+$. Let V_2 be a maximal set of attributes such that $V_2 \subseteq V$ and $V_2 \subseteq N$. By Union and Definition 3 of closure, $X \mapsto X^+$. Therefore, by Left Augmentation and Reduce, $XV_2 \mapsto A$. Since $N = \{A \mid A \in X \text{ and } A \notin X^+\}$, $V_2 \subseteq X$. Hence, $X \mapsto A$, which is a contradiction.

Our remaining proof obligation is to show that any AMFD not inferable from the set of AMFDs F with our axioms ($F \nvdash X \mapsto Y$) is not true ($F \not\models X \mapsto Y$). Suppose it is satisfied ($F \models X \mapsto Y$). It follows by the construction of table $\mathbf{t}$ that $Y \subseteq X^+$; otherwise, two rows of table $\mathbf{t}$ agree or are similar on X but disagree on some attribute A from Y. Since $Y \subseteq X^+$, by Lemma 3 it can be inferred that $X \mapsto Y$, which is a contradiction. Thus, whenever $X \mapsto Y$ does not follow from F by the AMFD axioms, F does not logically imply $X \mapsto Y$. That is, the axiom system is complete over AMFDs, which ends the proof . □

3.2 Discussion

The axiomatization for AMFDs is more involved than its FDs counterpart. A sound and complete axiomatization for traditional FDs consists of only three axioms: Reflexivity, Augmentation and Transitivity. Interestingly, *Reflexivity* (if $Y \subseteq X$, then $X \mapsto Y$) is not necessary true for AMFDs.

Example 2. (lack of Reflexivity) Consider table $\mathbf{t}$ (Table 4). Assume again that the values a and b are not equal ($a \neq b$) but they are similar ($a \approx_{m,\theta_A} b$) for each attribute A in N. Let attributes $\{BCD\} \subseteq N$. Therefore, the AMFDs $BCD \mapsto BCD$ and $BCD \mapsto BC$ are not satisfied in $\mathbf{t}$ because the values are similar on the left hand side of the dependencies, but not equal on their right hand side.

Table 5: Comparison of Axiomatizations

Dependency Class	Axioms
DDs [16]	Extended Reflexivity, Extended Augmentation, Extended Transitivity, Impropriety
SDs [11]	N\A
PODs [10]	Reflexivity, Augmentation, Transitivity, Reversal, Disjunction, Total Order, Impropriety
LODs [17, 19]	Reflexivity, Transitivity, Augmentation, Suffix, Normalization, Chain
MDs [7, 5, 9]	9 sound axioms (out of 11) appear in [9]
limited AMDs [this paper]	Void, Transitivity, Composition, Decomposition, Reduce
AMFDs [this paper]	Void, Transitivity, Composition, Decomposition, Reduce, Limited Reflexivity
CMFDs [13]	Footnote 5
FDs [3, 4, 12]	Reflexivity, Augmentation, Transitivity

Similarly, Augmentation, which is another axiom for FDs, does not necessary hold for AMFDs. (Augmentation states that if $X \mapsto Y$ then $XZ \mapsto YZ$.)

We replaced Reflexivity with Void and Limited Reflexivity in the axiomatization for AMFDs. Lack of Augmentation forced us to add Composition and Decomposition to the axiomatization. Note that Left Augmentation (Theorem 1) holds for AMFDs. Since Reflexivity does not hold for AMFDs, we had to add Reduce. Only Transitivity (base axiom for FDs) was preserved in axiomatization.

We also studied an axiomatization for a simplified version of MDs over a single table, rather than multiple tables as originally defined in [5, 7, 9]. We call these limited AMDs. The main difference between limited AMDs and AMFDs is that the former allow arbitrary similarity functions while the latter employ thresholds on similarity metrics. A sound and complete axiomatization for limited AMDs consists of the following five axioms: Void, Transitivity, Composition, Decomposition and Reduce. The proof will appear in the extended version of this paper; it is a simplified version of the proof of Theorem 1. In comparison to AMFDs, Limited Reflexivity does not hold for limited AMDs. A sound and complete axiomatization for a full class of MDs is more complex (Footnote 6), as it incorporates axioms that allow us to reason over multiple relations.

Table 5 compares the axiomatization of AMFDs and AMDs with other dependency classes. We point out several interesting observations below.

In contrast to MDs and AMFDs, the distance (gap) functions for DDs are defined at the dependency level for each attribute instead of the schema level. Therefore, Transitivity for DDs additionally requires an order relation over differential functions. For instance, if we have the dependency "if the date difference for two tuples is $\leq$ 30 days, then price $\geq$ \$50", then the dependency "if the date difference for two tuples is $\leq$ 30 days, then price $\geq$ \$40" must also hold.

There is an extra axiom for DDs (Impropriety) that accommodates inconsistencies between dependencies (this problem does not arise in AMFDs and MDs). For example, the following two dependencies are inconsistent since it is not pos-

Algorithm 1 Inference procedure for AMFDs

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

1: $F_{unused} \leftarrow F$; $n \leftarrow 0$
2: $X^n \leftarrow W$ where $W = \{A \mid A \in X \text{ and } \theta_A = 0\}$
3: **loop**
4: **if** $\exists\ V \mapsto Z \in F_{unused}$ and $V \subseteq \{X^n \cup X\}$ **then**
5: $X^{n+1} \leftarrow X^n \cup Z$
6: $F_{unused} \leftarrow F_{unused} - \{V \mapsto Z\}$
7: $n \leftarrow n + 1$
8: **else**
9: **return** X^n
10: **end if**
11: **end loop**

sible to instantiate a relation that satisfies both of them: a) if the date difference for two tuples is ≤ 30 days, then price = \$50; and b) if the date difference for two tuples is ≤ 30 days, then price $>$ \$50. Similarly, Augmentation and Reflexivity have to be modified for DDs to accommodate different distance functions used by different dependencies on the same attribute. For instance, different distance functions for the same attribute may result in the same actual distance.

Interestingly, as we traverse the hierarchy of dependencies, the number of axioms does not necessary decrease. There are 6 axioms for AMFDs, 7 for PODs and 6 for LODs versus 4 for DDs; however, there are 3 axioms for FDs at the bottom of hierarchy. There are two reasons for this. First, the axioms for DDs are quite complex. Second, as we go down the hierarchy, the dependencies become more specialized and therefore we may need more axioms to express their restricted semantics, e.g., lack of Reflexivity. As the dependencies become more generalized, some axioms must be weakened, e.g., Limited Reflexivity.

4 Inference Procedure

A goal of a dependency theory is to develop algorithms for the inference problem. Inference for DDs is co-NP-complete [16] and for MDs it is quadratic [7]. Since DDs and MDs generalize AMFDs, this sets an upper bound for the complexity of inference for AMFDs. However, computing closure, X^+, for AMFDs can be done more efficiently. It takes time proportional to the length of the dependencies in F, written out (linear time), which is as efficient as for FDs. (The complexity of inference for limited AMDs is also linear; the proof will appear in the extended version of this paper.) Algorithm 1 presents an inference procedure for AMFDs. Our experiments have shown that it is efficient. For 10 AMFDs prescribed over a dataset generated by the UIS Database [2], the algorithm runs in time $\leq$ 1ms.

Example 3. (inference) Let $F = \{AB \mapsto C,\ ABC \mapsto EG,\ EG \mapsto H\}$ denote the set of AMFDs. Also, let $\theta_C = 0$ and $\theta_D > 0$ for all attributes D in $ABEGH$ Let

us calculate the closure of set of attributes AB with Algorithm 1:

1) $X^0 = \{\}$; 2) $X^1 = C$; 3)$X^2 = CEG$; 4) $X^3 = CEGH$.

The closure of AB is $CEGH$. For traditional FDs, the closure of AB is $ABCEGH$.

Theorem 2. *(inference) Alg. 1 correctly computes the closure X^+ over AMFDs.*

Proof. First we show by induction on k that if Z is placed in X^k in Algorithm 1, then Z is in X^+.
Base case: $k = 0$. By Limited Reflexivity, $X \mapsto W$, where $W = \{A \mid A \in X$ and $\theta_A = 0\}$.
Induction step: $k > 0$. Assume that X^{k-1} only consists of the attributes in X^+. Suppose Z is placed in X^k because $VW \mapsto Z \in F_{unused}$, such that $V \subseteq X^{k-1}$ and $W \subseteq X$. Since $V \subseteq X^{k-1}$, we know by the induction hypothesis that $V \subseteq X^+$. Hence, by Lemma 3, $X \mapsto V$. Therefore, since $XV \mapsto VZ$ by Composition, then by Reduce and Decomposition $X \mapsto Z$. Thus, Z is in X^+.

Now we prove the opposite: if Z is in X^+, then Z is in the set returned by Algorithm 1. Suppose Z is in X^+ but Z is not in the set returned by Algorithm 1. Consider table $\mathbf{t}$ similar to that in Table 4. Table $\mathbf{t}$ has two tuples that agree on attributes in X^n, are similar but not equal on attributes X that are not subset of X^n, and disagree on all other attributes. We claim that $\mathbf{t}$ satisfies F. If not, let $P \mapsto Q$ be a dependency in F that is violated by $\mathbf{t}$. Then $P \subseteq X^n \cup X$ and Q cannot be a subset of $X^n \cup X$, if the violation happens. We used a similar argument in the proof of Theorem 1. Thus, by Algorithm 1, Lines 4–7, there exists X^{n+1}, which is a contradiction. □

5 Conclusions and Future Work

In this paper, we developed a hierarchy of dependency classes and laid out the theoretical foundations for AMFDs, which generalize traditional FDs. In future work, we plan to investigate the following problems.

- Determining whether a given AMFD holds on a given relation, and using AMFDs for data cleaning, similarly to how FDs were employed in previous data cleaning work [6, 7].
- Algorithms for automatic discovery of dependencies have been proposed for some dependencies, such as FDs and CFDs [8]. Similarly, we plan to study algorithms for discovering AMFDs.
- We plan to explore an inference framework for multiple dependencies. For example, the following inference rules hold: a) if an AMFD $X \mapsto Y$, then a CMFD $X \mapsto Y$; b) if an AMFD $X \mapsto Y$, then an FD $X \rightarrow Y$; c) if an FD $X \rightarrow Y$, then a CMFD $X \mapsto Y$. These rules along with the axioms for AMFDs (Figure 2) and CMFDs (Footnote 5), are *sound* for the integrated inference problem with FDs, AMFDs and CMFDs. However, an open question is if this rule set is *complete* and what is the complexity of the inference problem.
- Integrity constraints have been widely used in query optimization. For instance, FDs and LODs have been shown to be useful in simplifying queries

with *group by* and *order by* [14, 18, 17, 19] We believe that AMFDs can be used in similar ways to simplify SQL queries with similarity operators [15].

References

1. Economist Intelligence Unit analysis, http://www.economistinsights.com/technology-innovation/analysis/big-data.
2. UIS Data Generator, http://www.cs.utexas.edu/users/ml/riddle/data.html.
3. W. Armstrong. Dependency Structures of Database relationships. In *Proceedings of the IFIP Congress*, pages 580–583, 1974.
4. C. Beeri and P. Bernstein. Computional Problems Related to the Design of Normal Form Relational Schemas. *TODS 4(1):*, 4(1):30–59, 1979.
5. L. Bertossi, S. Kolahi, and V. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. In *ICDT*, pages 268–279, 2011.
6. G. Beskales, I. F. Ilyas, and L. Golab. Sampling the repairs of functional dependency violations under hard constraints. *PVLDB*, 3(1):197–207, 2010.
7. W. Fan. Dependencies Revisited for Improving Data Quality. In *PODS*, pages 159–170, 2008.
8. W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering Conditional Functional Dependencies. *TKDE*, 23(5):683–698, 2011.
9. W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about Record Matching Rules. *PVLDB*, 2(1):407–418, 2009.
10. S. Ginsburg and R. Hull. Order dependency in the relational model. *Theoretical Computer Science*, 26(1–2):149–195, 1983.
11. L. Golab, H. Karloff, F.Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1):574–585, 2009.
12. Y. Huhtala, J. Karkk, P. Porkka, and H. Toivonen. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Computer Journal*, 42(2):100–111, 1999.
13. N. Koudas, A. Saha, A. Srivastava, and S. Venkatasubramanian. Metric Functional Dependencies. In *ICDE, 1291-1294*, 2009.
14. M. Malkemus, P. S., B. Bhattacharjee, L. Cranston, T. Lai, and F. Koo. Predicate Derivation and Monotonicity Detection in DB2 UDB. In *ICDE, 939-947*, 2005.
15. Y. N. Silva and S. Pearson. Exploiting database similarity joins for metric spaces. *PVLDB*, 5(12):1922–1925, 2012.
16. S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *TODS*, 36(3):16, 2011.
17. J. Szlichta, P. Godfrey, and J. Gryz. Fundamentals of Order Dependencies. *PVLDB*, 5(11):1220–1231, 2012.
18. J. Szlichta, P. Godfrey, J. Gryz, W. Ma, P. Pawluk, and C. Zuzarte. Queries on dates: fast yet not blind. In *EDBT 497-502*, 2011.
19. J. Szlichta, P. Godfrey, J. Gryz, and C. Zuzarte. Expressiveness and Complexity of Order Dependencies. *PVLDB 6(14): 1858-1869*, 2013.
20. M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller. Continuous data cleaning. In *ICDE*, pages 244–255, 2014.