Tractable vs. Intractable Cases of Query Answering under Matching Dependencies

Leopoldo Bertossi
Carleton University, SCS
Ottawa, Canada

Jaffer Gardezi
University of Ottawa, SITE.
Ottawa, Canada

Abstract. Matching Dependencies (MDs) are a recent proposal for declarative entity resolution. They are rules that specify, on the basis of similarities satisfied by values in a database, what values should be considered duplicates, and have to be matched. On the basis of a chase-like procedure for MD enforcement, we can obtain clean (duplicate-free), and possibly several, *resolved* instances. The *resolved answers* to a query are invariant under the class of resolved instances. Previous work identified classes of queries and sets of MDs for which resolved query answering is tractable, with special emphasis on cyclic sets of MDs. In this work we further investigate the complexity of this problem, identifying intractable cases, and exploring the frontier between tractability and intractability. We concentrate mostly on acyclic sets of MDs. For a special case we obtain a dichotomy result relative to *NP*-hardness.

1 Introduction

A database may contain several representations of the same external entity, i.e. "duplicates", which may be undesirable; and the database has to be cleaned. The problem of *duplicate- or entity-resolution* (ER) is about (a) detecting duplicates, and (b) merging duplicate representations into single representations. It is a classic and complex problem in data management and data cleaning, in particular [7, 9, 3]. In this work we deal with the merging part of the problem, in a relational context.

The problem can be approached by specifying what attribute values have to be matched (made identical) under what conditions. For this, a declarative language with a precise semantics can be used. In this direction, matching dependencies (MDs) have been recently introduced [10, 11]. They represent rules for resolving pairs of duplicate representations (two tuples at a time). When certain similarity relationships between attribute values hold, an MD indicates what attribute values have to be made the same.

Example 1. The similarities of phone and address indicate that the tuples refer to the same person, and the names should be matched. Here, 723-9583 $\approx$ (750) 723-9583 and 10-43 Oak St. $\approx$ 43 Oak St. Ap. 10.

People	Name	Phone	Address
	John Smith	723-9583	10-43 Oak St.
	J. Smith	(750) 723-9583	43 Oak St. Ap. 10

This MD captures this resolution policy: (with P standing for predicate $People$): $P[Phone] \approx P[Phone] \wedge P[Address] \approx P[Address] \rightarrow P[Name] \doteq P[Name]$. It involves only one database predicate, but an MD may involve two different relations. We can also have several, interacting MDs on the schema. □

The framework for MD-based ER we use was introduced in [12], with a precise, chase-based semantics for the MDs originally introduced in [11]. The problem of *resolved query answering* (RQA) was introduced in [12]. For a fixed set of MDs, and a fixed

query, it is about deciding, given an "unresolved" instance, and a candidate query answer $\bar{a}$, whether $\bar{a}$ is an answer to the query under all admissible ways of resolving the duplicates as dictated by the MDs. This problem is generally intractable [12].

The RQA problem was studied further in [14, 13]. A class of tractable cases of RQA was identified [14], for which a technique based on query rewriting into stratified Datalog with aggregation was developed [13]. In those tractable cases, we find conjunctive queries with certain restrictions on joins, and sets of MDs that *cyclically* depend on each other. These are the (cyclic) *HSC sets* identified in [14]. It was shown that, in general, cyclic dependencies on MDs make the problem tractable, because the requirement of chase termination imposes a relatively simple structure on the clean instances [14].

We concentrate here on *acyclic sets* of MDs, which completely change the picture wrt. previous work. As just mentioned, for HSC sets, tractability of RQA holds [14]. This is the case, e.g., for the cyclic $M = \{R[A] \approx R[A] \rightarrow R[B] \doteq R[B],\ R[B] \approx R[B] \rightarrow R[A] \doteq R[A]\}$. However, as we will show, for the following acyclic, somehow syntactically similar example, $M' = \{R[A] \approx R[A] \rightarrow R[B] \doteq R[B],\ \ R[B] \approx R[B] \rightarrow R[C] \doteq R[C]\}$, RQA can be intractable. This example, and our general results, show that, possibly counter-intuitively, the presence of cycles in sets of MDs tends to make resolved query answering easier in comparison with the acyclic case.

We further explore the complexity of RQA. Instead of considering isolated intractable cases as in [12, 14], we take a more systematic approach, developing syntactic criteria on sets of two MDs that, when satisfied by a given pair of MDs, implies intractability of RQA. We show, under an additional assumption about the similarity operators, that RQA is tractable for sets of MDs not satisfying these criteria, leading to a dichotomy result. We extend these results also considering (in)tractability of sets of more than two MDs. All the results apply to acyclic sets of MDs, and are complementary to those in [14, 13], providing a broader picture of the complexity of RQA.

Summarizing, in this paper, we undertake a systematic investigation of the data complexity of the problems of deciding and computing resolved answers to conjunctive queries under MDs. This sheds light on the intrinsic computational limitations of retrieving, from a database with unresolved duplicates, the information that is invariant under the ER process as captured by MDs. Our contributions are the following:

1. We identify a class of conjunctive queries that are relevant for the investigation of tractability vs. intractability of RQA. Intuitively, these queries return data that can be modified by application of the MDs. We call them *changeable attribute queries*.

2. Having investigated in [13, 14] cases of cyclic sets of MDs, we complement these results by studying the complexity of RQA for sets of MDs that do not have cycles.

3. For certain *pairs* of MDs that satisfy a syntactic condition, we establish an intractability result, proving that deciding resolved answers to changeable attribute queries is $\mathcal{NP}$-hard in data.

4. For similarity relations that are transitive (a special case), we establish that the conditions for hardness mentioned in the previous item, lead to a dichotomy result: pairs of MDs that satisfy them are always *hard*, otherwise they are always *easy* (for RQA). This shows, in particular, that the result mentioned in item 3. cannot be extended to a wider class of MDs for arbitrary similarity relations. We also prove that the dichotomy result does not hold when the hypothesis on similarity is not satisfied.

5. Relying on the results for pairs of MDs, we consider acyclic *sets* of MDs of arbitrary size. In particular, we prove intractability of the RQA problem for certain acyclic sets of MDs that have the syntactic property of *non-inclusiveness*.

The structure of the paper is as follows. Section 2 introduces notation, terminology, and previous results. Section 3 identifies classes of MDs, queries and assumptions that are relevant for this research. Sections 3.1 and 4 investigate the complexity of the problem of computing resolved answers for sets of two MDs. Section 5 extends those results to sets of MDs of arbitrary size. Section 6 summarizes results, and makes comparisons with *consistent query answering*. Full proofs of our results can be found in [6].

2 Preliminaries

In this work we consider relational database schemas and instances. Schemas are usually denoted with $\mathcal{S}$, and contain relational predicates. Instances are usually denoted with D. Matching dependencies (MDs) are symbolic rules of the form:

$$\bigwedge_{i,j} R[A_i] \approx_{ij} S[B_j] \;\rightarrow\; \bigwedge_{k,l} R[A_k] \doteq S[B_l], \tag{1}$$

where R, S are relational predicates in $\mathcal{S}$, and the $A_i, ...$ are attributes for them. The LHS captures similarity conditions on a pair of tuples belonging to the extensions of R and S in an instance D. We abbreviate (1) as: $R[\bar{A}] \approx S[\bar{B}] \;\rightarrow\; R[\bar{C}] \doteq S[\bar{E}]$.

The similarity predicates (or operators) $\approx$ are domain-dependent and treated as built-ins. For them we assume symmetry and reflexivity, but not transitivity.

MDs have a *dynamic interpretation* requiring that those values on the RHS should be updated to some arbitrary value in common (of the database domain). Attributes on a RHS of an MD are called *changeable*. MDs are expected to be "applied" iteratively until duplicates are solved, through *chase sequences*. In order to keep track of the changes and comparing tuples and instances, we use global tuple identifiers, a non-changeable surrogate key for each database predicate. The auxiliary, extra attribute (when shown) appears as the first attribute in a relation, e.g. t is the identifier in $R(t, \bar{x})$. A *position* is a pair (t, A) with t a tuple id, and A an attribute (of the relation where t is an id). The *position's value*, $t[A]$, is the value for A in tuple (with id) t.

2.1 MD semantics

The semantics of MDs acting on database instances [12] is based on a *chase procedure* that is iteratively applied to the original instance D. A *resolved instance* D' is obtained from a finitely terminating sequence of database instances:

$$D =: D_0 \;\mapsto\; D_1 \;\mapsto\; D_2 \;\mapsto\; \cdots \;\mapsto\; D_n =: D'. \tag{2}$$

D' satisfies the MDs as *equality generating dependencies* [1], i.e. replacing $\doteq$ by $=$.

The semantics specifies the one-step transitions or updates applied to go from D_{i-1} to D_i, i.e. "$\mapsto$" in (2). Only *modifiable positions* within the instance are allowed to change their values in such a step, and as forced by the MDs. Actually, the modifiable positions syntactically depend on the set M of MDs and the instance at hand; and can be recursively defined (see [12, 14] for the details).[1] Intuitively, a position (t, A) is modifiable iff: (a) There is a t' such that t and t' satisfy the similarity condition of an MD with A on the RHS; and (b) $t[A]$ has not already been resolved (it is different from one of its other duplicates).

[1] As a consequence, a *changeable* attribute may not necessarily give rise to a corresponding *modifiable* position for a given instance at hand.

Example 2. For schema $R(A, B)$, consider the MD $R[A] = R[A] \rightarrow R[B] \doteq R[B]$, and the instance $R(D)$ below. The positions of the underlined values in D are modifiable, because their values are unresolved (wrt the MD and instance $R(D)$).

$R(D)$	A	B
t_1	a	$\underline{b}$
t_2	a	$\underline{c}$

$\mapsto$

$R(D')$	A	B
t_1	a	d
t_2	a	d

D' is a resolved instance since it satisfies the MD interpreted as the FD $R : A \rightarrow B$. Here, the update value d is arbitrary.

D' has no modifiable positions with unresolved values: the values for B are already the same, so there is no reason to change them (and we don't). □

More formally, the *single step semantics* ($\mapsto$ in (2)) is as follows. Each pair (D_i, D_{i+1}) in an update sequence (2), i.e. a chase step, must *satisfy* the set M of MDs *modulo unmodifiability*, denoted $(D_i, D_{i+1}) \models_{um} M$, which holds iff: (a) For every MD in M, say $R[\bar{A}] \approx S[\bar{B}] \rightarrow R[\bar{C}] \doteq S[\bar{D}]$ and pair of tuples t_R and t_S, if $t_R[\bar{A}] \approx t_S[\bar{B}]$ in D_i, then $t_R[\bar{C}] = t_S[\bar{D}]$ in D_{i+1}; and (b) The value of a position can only differ between D_i and D_{i+1} if it is modifiable wrt D_i. Accordingly, in (2) we also require that $(D_i, D_i) \not\models_{um} M$, for $i < n$, and $(D_n, D_n) \models_{um} M$ (the *stability* condition).[2]

This semantics captures as close as possible the spirit of MDs as originally, and rather informally introduced in [11], and also *uncommitted* in the sense that the MDs do not specify how the matchings have to be realized (also as in [11]).

Example 3. Consider the instance $R(D)$ below and the set of MDs: $\{R[A] = R[A] \rightarrow R[B] \doteq R[B];\ R[B] = R[B] \rightarrow R[C] \doteq R[C]\}$. Attribute $R(C)$ is changeable. Position (t_2, C) is not modifiable wrt. M and D: There is no justification to change its value *in one step* on the basis of an MD and D. Position (t_1, C) is modifiable. D has two resolved instances, $R(D_1)$ and $R(D_2)$. $R(D_1)$ cannot be obtained in a single (one step) update (the underlined value is for a non-modifiable position). But $R(D_2)$ can.

$R(D)$	A	B	C
t_1	a	b	d
t_2	a	c	$\underline{e}$
t_3	a	b	e

$R(D_1)$	A	B	C
t_1	a	b	d
t_2	a	b	d
t_3	a	b	d

$R(D_2)$	A	B	C
t_1	a	b	e
t_2	a	b	e
t_3	a	b	e

□

For arbitrary sets of MDs, some (admissible) chase sequences may not terminate. However, it can be proved that there are always terminating chase sequences. As a consequence, for some sets of MDs, there are both terminating and non-terminating chase sequences. In any case, the class of resolved instances is always well-defined.

We prefer *resolved instances* that are the closest to the original instance. A *minimally resolved instance* (MRI) of D is a resolved instance D' whose *number of changes of attribute values* wrt. D is a minimum. In Example 3, instance D_2 is an MRI, but not D_1 (2 vs. 3 changes). We denote with $Res(D, M)$ and $MinRes(D, M)$ the classes of resolved, resp. minimally resolved, instances of D wrt M.

Infinite chase sequences may occur when the MDs *cyclically* depend on each other, in which case updated instances in a such a sequence may alternate between two or more states [14, Example 6]. However, for the chase sequences that do terminate in a minimally resolved instance, the chase imposes a relatively easily characterizable structure [14, 13], allowing us to obtain a query rewriting methodology. So, cycles help us achieve tractability for some classes of queries [13] (cf. Section 2.2).

[2] The case $D' = D_0$ occurs only when D is already resolved.

On the other side, it has been shown that if a set of MDs satisfies a certain *acyclicity* property, then all chase sequences terminate after a number of iterations that depends only on the set of MDs and not on the instance [12, Lemma 1] (cf. Theorem 1 below). But the number of resolved instances may still be "very large". Sets of MDs considered in this work are acyclic.

2.2 Resolved query answers

Given a conjunctive query $\mathcal{Q}$, a set of MDs M, and an instance D, the *resolved answers* to $\mathcal{Q}$ from D are invariant under the entity resolution process, i.e. they are answers to $\mathcal{Q}$ that are true in all MRIs of D:

$$ResAns_M(\mathcal{Q}, D) := \{\ \bar{a}\ \mid D' \models \mathcal{Q}[\bar{a}],\ \text{for every } D' \in MinRes(D, M)\}. \quad (3)$$

The *resolved query answering* (RQA) corresponding decision problem is $RA(\mathcal{Q}, M) := \{(D, \bar{a}) \mid \bar{a} \in ResAns_M(\mathcal{Q}, D)\}$.

In [13, 14], a query rewriting methodology for computing resolved answers to queries under MDs was presented. In this case, the rewritten queries turn out to be Datalog queries with counting, and can be obtained for two main kinds of sets of MDs: (a) MDs do not depend on each other, i.e. *non-interacting* sets of MDs [12]; (b) MDs that depend cyclically on each other, e.g. $R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$ and $R[B] \approx R[B] \rightarrow R[A] \doteq R[A]$ (or relationships like this by transitivity).

For these sets of MDs, a conjunctive query can be rewritten to retrieve, in polynomial time in data, the resolved answers, provided the queries have no joins on existentially quantified variables corresponding to changeable attributes. The latter form the class of *unchangeable attribute join conjunctive* (UJCQ) queries [14].

For example, for the MD $R[A] = R[A] \rightarrow R[B, C] \doteq R[B, C]$ on schema $R[A, B, C]$, $\mathcal{Q}$: $\exists x \exists y \exists z (R(x, y, c) \wedge R(z, y, d))$ is *not* UJCQ; whereas $\mathcal{Q}'$: $\exists x \exists z (R(x, y, z) \wedge R(x, y', z')$ is UJCQ. For queries outside UJCQ, the resolved answer problem can be intractable even for one MD [14].

The set of MDs (4), which is neither non-interacting nor cyclic, is not covered by the positive cases for Datalog rewriting above.

$$\begin{aligned} R[A] &\approx R[A] \rightarrow R[B] \doteq R[B], \\ R[B] &\approx R[B] \rightarrow R[C] \doteq R[C], \end{aligned} \quad (4)$$

Actually, for this set RQA becomes intractable for very simple queries, like $\mathcal{Q}(x, z) : \exists y R(x, y, z)$, that is UJCQ [12]. Sets of MDs like (4) are the main focus of this work.

3 Intractability of RQA

We just briefly described classes of queries and MDs for which RQA can be done in polynomial time in data (via the Datalog rewriting). We showed that there are intractable cases, by pointing to a specific query and set of MDs. Natural questions that we start addressing are: (a) What is the complexity of RQA outside the Datalog rewritable cases? (b) Do the exhibited query and MDs fall into a more general intractable class?

For all sets M of MDs we consider below, *we assume that at most two relational predicates, say R, S, appear in M*, e.g. as in $M = \{R[A] \approx S[B] \rightarrow R[C] \doteq S[E]\}$. In same cases we assume that there are exactly two predicates. The purpose of this restriction is to simplify the presentation. All results can be generalized to sets of MDs with more than two predicates. To do this, definitions and conditions concerning the two relations in the MDs can be extended to cover the additional relations as well.

At the other extreme, when a single predicate occurs in M, say R, as in Example 3, the results for at most two predicates can be reformulated and applied by replacing S with R'. Although R and R' are the same relation in this case, the prime is used to distinguish between the two tuples to which the MD refers.

All the sets of MDs considered below are both *interacting* (non-interaction does not bring complications) and *acyclic*. Both notions and others can be captured in terms of the MD *graph*, $MDG(M)$, of M. It is a directed graph, such that, for $m_1, m_2 \in M$, there is an edge from m_1 to m_2 if there is an overlap between $RHS(m_1)$ and $LHS(m_2)$ (the right- and left-hand sides of the arrows as sets of attributes) [12]. Accordingly, M *is acyclic when* $MDG(M)$ *is acyclic*. In fact, the sets of MDs in this work satisfy a stronger property, defined below, which we call *strong acyclicity*.

Definition 1. [12] 1. Let M be a set of MDs on schema $\mathcal{S}$. (a) The symmetric binary relation $\doteq_r$ relates attributes $R[A]$, $S[B]$ of $\mathcal{S}$ whenever there is $m \in M$ in which $R[A] \doteq S[B]$ occurs. (b) The *attribute closure* of M is the reflexive and transitive closure of $\doteq_r$. (c) $E_{R[A]}$ denotes the equivalence class of attribute $R[A]$ in the attribute closure of M.
2. The *augmented MD-graph* of M, denoted $AMDG(M)$, is a directed graph with a vertex labeled with m for each $m \in M$, and with an edge from m to m' iff there is an attribute, say $R[A]$, with $R[A] \in RHS(m)$ and $E_{R[A]} \cap LHS(m') \neq \emptyset$.
3. M is *strongly acyclic* if $AMDG(M)$ has no cycles. □

Because $R[A] \in E_{R[A]}$, for any set M of MDs, all edges in $MDG(M)$ are also edges in $AMDG(M)$. So, strong acyclicity implies acyclicity, but the converse is not true.

Example 4. The set M of MDs $m_1: \ R[F] \approx S[G] \rightarrow R[A] \doteq S[H]$; $m_2: \ R[A] \approx S[B] \rightarrow R[C] \doteq S[E]$; $m_3: \ R[C] \approx S[E] \rightarrow R[I] \doteq S[H]$ is acyclic but not strongly acyclic. $MDG(M)$ has three vertices, m_1, m_2, m_3, and edges (m_1, m_2) and (m_2, m_3). $AMDG(M)$ has the additional edge (m_3, m_2), because $E_{R[I]} = \{R[I], S[H], R[A]\} \cap LHS(m_2) = \{R[A]\}$. □

In this work, we consider strongly acyclic sets of MDs. In particular, two interesting and common kinds that form large classes of sets M of MDs: *linear pairs*, which consist of two MDs such that $MDG(M)$ contains a single edge from one to the other (c.f. Definition 5); and acyclic sets that are *pair-preserving* (c.f. Definition 7). From the definitions of these two kinds of sets of MDs it will follow that they are strongly acyclic.

Theorem 1. [12] Let M be a strongly acyclic set of MDs on schema $\mathcal{S}$, and D an instance for $\mathcal{S}$. Every sequence of M-based updates to D as in (2) terminates with a resolved instance after at most $d + 1$ steps, where d is the maximum length of a path in $AMDG(M)$. □

As mentioned previously, there may be infinite chase sequences when M is not acyclic. For the cyclic case, Theorem 1 only tells us about the chase termination and lengths, but not about the data; and does not guarantee tractability for RQA. This leaves room for tractable and intractable cases. Actually, it can still be the case that there are exponentially many MRIs. A reason for this is that the application of an MD to an instance may produce new similarities among the values of attributes in $RHS(m_1)$ that are not strictly required by the chase, but result from a particular choice of update values. Such "accidental similarities" affect subsequent updates, resulting in exponentially many possible update sequences. This is illustrated in the next example.

Example 5. Consider the strongly acyclic set M: $R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$; $R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$. When the following instance is updated according to M, the sets of value positions $\{t_1[B], t_2[B]\}$ and $\{t_3[B], t_4[B]\}$ must be merged.

$R(D_1)$	A	B	C
t_1	a	m	e
t_2	a	d	f
t_3	b	c	g
t_4	b	k	h

$\mapsto$

$R(D_1)$	A	B	C
t_1	a	m	e
t_2	a	m	f
t_3	b	m	g
t_4	b	m	h

One possible update is as above. The similarities between the attribute B values of the top and bottom pairs of tuples are accidental, because they result from the choice of update values. In the absence of accidental similarities, there is only one possible set of sets of values that are merged in the second update, namely $\{\{t_1[C], t_2[C]\}, \{t_3[C], t_4[C]\}\}$.

Accidental similarities increase the complexity of query answering over the instance by adding another possible set of sets of merged values, $\{\{t_1[C], t_2[C], t_3[C], t_4[C]\}\}$. More generally, for an instance with n sets of merged value positions in the B column, the number of possible sets of sets of value positions in the C column that are merged in the second update is $\Omega(2^{n^2})$. □

We want to investigate the frontier between tractability and intractability. For this reason, we make the assumption that, *for each similarity relation, $\approx$, there is an infinite set of mutually dissimilar values*. Actually, without this assumption, the resolved answer problem becomes immediately tractable for certain similarity operators (e.g. transitive similarity operators). This is because, for these operators, the whole class of minimal resolved instances of an instance can be computed in polynomial time.

Proposition 1. For strongly acyclic sets of MDs, if the similarity predicates are transitive and there is no infinite set of mutually dissimilar values, the set of minimal resolved instances for a an instance D can be computed in polynomial time in the size of D. □

Our next results require some terms and notation that we now introduce.

Definition 2. For a set M of MDs with predicates R and S, a *changeable attribute query* $\mathcal{Q}$ is a (conjunctive) query in UJCQ, containing a conjunct of the form $R(\bar{x})$ or $S(\bar{y})$ all whose variables are free and none occur in another conjunct of the form $R(\bar{x})$ or $S(\bar{y})$. Such a conjunct is a *join-restricted free occurrence* of the predicate R or S.□

By definition, the class of *changeable attribute queries* (CHAQ) is a subclass of UJCQ. Both depend on the set of MDs at hand. For example, for the MDs in (4), $\exists y R(x, y, z) \in$ UJCQ $\setminus$ CHAQ, but $\exists w \exists t(R(x, y, z) \wedge S(x, w, t)) \in$ CHAQ. We confine our attention to UJCQ and subsets of it, because, as mentioned in the previous section, intractability limits the applicability of the duplicate resolution method for queries outside UJCQ.

The requirement that the query contains a join-restricted free occurrence of R or S eliminates from consideration certain queries in UJCQ for which the resolved answer problem is immediately tractable. For example, for the MDs in (4), the query $\exists y \exists z R(x, y, z)$ is not CHAQ, and is tractable simply because it does not return the values of a changeable attribute (the resolved answers are the answers in the usual sense). The restriction on joins simplifies the analysis while still including many useful queries.

In order to eliminate queries like $\exists y \exists z R(x, y, z)$ wrt M in (4), CHAQ imposes a strong condition. Actually, the condition can be weakened, requiring to have *at least*

one of the variables satisfying the condition in the definition for CHAQ. Weakening the condition makes the presentation much more complex since a finer interaction with the MDs has to be brought into the picture. (We leave this issue for an extended version.)

Definition 3. A set M of MDs is *hard* if, for every CHAQ $\mathcal{Q}$, $RA(\mathcal{Q}, M)$ is *NP*-hard. M is *easy* if, for every CHAQ $\mathcal{Q}$, $RA(\mathcal{Q}, M)$ is in *PTIME*. □

Of course, a set of MDs still may not be hard or easy. For the resolved answer problem, membership of *NP* is open. However, for strongly acyclic sets, the bound on the length of the chase implies an upper bound of Π_2^P [12, Theorem 5].

In the following we give some syntactic conditions that guarantee hardness for classes of MDs. To state them we need to introduce some useful notions first.

Definition 4. For an MD m, the symmetric binary relation $LRel(m)$ ($RRel(m)$) relates pairs of attributes $R[A]$ and $S[B]$ when $R[A] \approx S[B]$ (resp. $R[A] \doteq S[B]$) appears in m. An *L-component* (*R-component*) of m is an equivalence class of the reflexive, transitive closure $LRel(m)^{=+}$ (resp. $RRel(m)^{=+}$) of $LRel(m)$ (resp. $RRel(m)$). □

Example 6. For $R[A] \approx S[B] \wedge R[A] \approx S[C] \rightarrow R[E] \doteq S[F] \wedge R[G] \doteq S[H]$, there is only one L-component: $\{R[A], S[B], S[C]\}$; and two R-components: $\{R[E], S[F]\}$ and $\{R[G], S[H]\}$. □

3.1 Hardness of linear pairs of MDs

Most of the results that follow already hold for pairs of MDs.

Definition 5. A set $M = \{m_1, m_2\}$ of MDs is a *linear pair*, denoted (m_1, m_2), if its graph $MDG(M)$ consists of vertices m_1 and m_2 with only an edge from m_1 to m_2. □

Notice that if (m_1, m_2) is a generic linear pair, say

$$\begin{aligned} m_1\colon\ & R[\bar{A}] \approx_1 S[\bar{B}] \rightarrow R[\bar{C}] \doteq S[\bar{E}], \\ m_2\colon\ & R[\bar{F}] \approx_2 S[\bar{G}] \rightarrow R[\bar{H}] \doteq S[\bar{I}], \end{aligned} \tag{5}$$

then, from the definition of the MD graph, it follows that $(R[\bar{C}] \cup S[\bar{E}]) \cap (R[\bar{F}] \cup S[\bar{G}]) \neq \emptyset$, whereas $(R[\bar{H}] \cup S[\bar{I}]) \cap (R[\bar{A}] \cup S[\bar{B}]) = \emptyset$. In the following we have to analyze other different forms of (non-)interaction between the attributes in linear pairs.

Definition 6. Let (m_1, m_2) be a linear pair as in (5). (a) B_R is a binary (reflexive and symmetric) relation on attributes of R: $(R[U_1], R[U_2]) \in B_R$ iff $R[U_1]$ and $R[U_2]$ are in the same R-component of m_1 or the same L-component of m_2. Similarly for B_S. (b) An *R-equivalent set* (R-ES) of attributes of (m_1, m_2) is an equivalence class of $TC(B_R)$, the transitive closure of B_R, with at least one attribute in the equivalence class belonging to $LHS(m_2)$. The definition of an *S-equivalent set* (S-ES) is similar, with R replaced by S. (c) An (R or S)-ES E of (m_1, m_2) is *bounded* if $E \cap LHS(m_1)$ is non-empty. □

Example 7. Consider the schema $R[A, C, F, H, I, M], S[B, D, E, G, N]$, and the linear pair (m_1, m_2) with:
$m_1\colon\ R[A] \approx S[B] \rightarrow R[C] \doteq S[D] \wedge R[C] \doteq S[E] \wedge R[F] \doteq S[G] \wedge R[H] \doteq S[G]$,
$m_2\colon\ R[F] \approx S[E] \wedge R[I] \approx S[E] \wedge R[A] \approx S[E] \wedge R[F] \approx S[B] \rightarrow R[M] \doteq S[N]$.

It holds: (a) $B_R(R[F], R[H])$ due to the occurrence of $R[F] \doteq S[G]$, $R[H] \doteq S[G]$. (b) $B_R(R[F], R[I])$ due to $R[F] \approx S[E]$, $R[I] \approx S[E]$. (c) $B_R(R[I], R[A])$ due to $R[I] \approx S[E]$, $R[A] \approx S[E]$. (d) $\{R[A], R[F], R[I], R[H]\}$ is an R-ES, and since $\{R[A], R[F], R[I], R[H]\} \cap LHS(m_1) = \{R[A]\} \neq \emptyset$, it is also bounded. □

Theorem 2. Let (m_1, m_2) be a linear pair, with relational predicates R and S. Let E_R, E_S be the sets of R-ESs and S-ESs, resp. The pair (m_1, m_2) is hard if $RHS(m_1) \cap RHS(m_2) = \emptyset$, and at least one of (a) and (b) below holds: (a) All of the following hold: (i) $Attr(R) \cap (RHS(m_1) \cap LHS(m_2)) \neq \emptyset$. (ii) There are unbounded ESs in E_R. (iii) For some L-component L of m_1, $Attr(R) \cap (L \cap LHS(m_2)) = \emptyset$. (b) Same as (a), but with R replaced by S. □

Theorem 2 says that a linear pair of MDs is hard unless the syntactic form of the MDs is such that there is a certain association between changeable attributes in $LHS(m_2)$ and attributes in $LHS(m_1)$ as specified by conditions (ii) and (iii).

For pairs of MDs satisfying the negation of (a)(ii) or that of (a)(iii) (or the negation of (b)(ii) or that of (b)(iii)) in Theorem 2, the similarities resulting from applying m_2 are restricted to a subset of those that are already present among the values of attributes in $LHS(m_1)$, making the problem tractable. However, when condition (ii) or (iii) is satisfied, accidental similarities among the values of attributes in $RHS(m_1)$ cannot be passed on to values of attributes in $RHS(m_2)$.

Example 8. The linear pair (m_1, m_2), with $m_1 : R[A] \approx S[B] \rightarrow R[C] \doteq S[D]$; $m_2 : R[C] \approx S[D] \rightarrow R[E] \doteq S[F]$, is hard. In fact, first: $RHS(m_1) \cap RHS(m_2) = \emptyset$.

Now, it satisfies condition (a): Condition (a)(i) holds, because $R[C] \in RHS(m_1) \cap LHS(m_2)$. Conditions (a)(ii) and (a)(iii) are trivially satisfied, because there are no attributes of $LHS(m_1)$ in $LHS(m_2)$. □

As mentioned above, Theorem 2 generalizes to the case of more or fewer than two database predicates. It is easy to verify, for the former case, that if there are more than two predicates in a linear pair, then there must be exactly three of them, one of which appears in both MDs. In this case, hardness is implied by condition (a) in Theorem 2 alone, with R the predicate in common.

Example 9. The linear pair (m_1, m_2) with three predicates $m_1 : R[A] \approx S[B] \rightarrow R[C] \doteq S[E]$; $m_2 : R[C] \approx P[B] \rightarrow R[F] \doteq P[G]$ is hard if it satisfies condition (a) in Theorem 2, which it does: (i) $Attr(R) \cap (RHS(m_1) \cap LHS(m_2)) = \{R[C]\}$. (ii) The ES $\{R[C]\}$ is unbound. (iii) Part (iii) holds with $L = \{R[A], S[B]\}$. □

With only one predicate R in the linear pair, in order to apply Theorem 2, we need to derive from it a special result, Corollary 1 below. It is obtained by first labeling the different occurrences of the (same) predicate in M, and then generating conditions (four of them, analogous to (a) and (b) in Theorem 2) for the labeled version, M'. When M' satisfies those conditions, the original set M is hard. An algorithm, *Conditions*, (described in detail in [6]) does both the labeling and the condition generation to be checked on M'. After the labeling, there is still only one predicate in M'. The labeling simply provides a convenient way to refer to different sets of attributes. Example 10 demonstrates in informal terms the use of the algorithm and the application of the corollary.

Corollary 1. A linear pair containing one predicate is hard if it satisfies $RHS(m_1) \cap RHS(m_2) = \emptyset$ and at least one of the four sets of three conditions (i)-(iii) generated by Algorithm *Conditions*. □

Example 10. Consider the linear pair M: $m_1 : R[A] \approx R[B] \wedge R[C] \approx R[E] \rightarrow R[F] \doteq R[G] \wedge R[B] \doteq R[G]$; $m_2 : R[G] \approx R[H] \wedge R[B] \approx R[I] \wedge R[L] \approx R[I] \rightarrow R[J] \doteq R[K]$. Algorithm *Conditions* produces the following labeling:

$$m_1' : \ R_1^1[A] \approx R_1^2[B] \wedge R_1^1[C] \approx R_1^2[E] \rightarrow R_1^1[F] \doteq R_1^2[G] \wedge R_1^1[B] \doteq R_1^2[G],$$
$$m_2' : \ R_2^1[G] \approx R_2^2[H] \wedge R_2^1[B] \approx R_2^2[I] \wedge R_2^1[L] \approx R_2^2[I] \rightarrow R_2^1[J] \doteq R_2^2[K].$$

With the above labeling, R^1 (R^2)-equivalent sets can be defined analogously to R (S)-equivalent sets in the two relation case, except that they generally include attributes from two "relations", R_1^1 and R_2^1 (R_1^2 and R_2^2), instead of one. For example, in $\{m_1', m_2'\}$, one R^1-ES is $\{R_1^1[F], R_1^1[B], R_2^1[B], R_2^1[L]\}$.

The conditions output by *Conditions* for the combination $X = 1$, $Y = 2$ is the following: (i) $Attr(R_1^2) \cap Attr(R_2^1) \cap (RHS(m_1) \cap LHS(m_2)) \neq \emptyset$, (ii) There are R^1-equivalent sets that do not contain attributes in $Attr(R_1^2) \cap LHS(m_1)$, and (iii) For some L-component L of m_1, $Attr(R_1^2) \cap Attr(R_2^1) \cap (L \cap LHS(m_2)) = \emptyset$.

These conditions are satisfied by M'. In fact, for (i) this set is $\{R_1^2[G]\}$; for (ii) the R^1-ES $\{R_2^1[G]\}$ satisfies the condition; and for (iii) $L = \{R_1^1[C], R_1^2[E]\}$ satisfies the condition. Thus, by Corollary 1, M is hard. □

Example 11. (example 5 cont.) M is hard by Corollary 1. In fact, Algorithm *Conditions* produces the following labeled set M': $R_1^1[A] \approx R_1^2[A] \rightarrow R_1^1[B] \doteq R_1^2[B]$; $R_2^1[B] \approx R_2^2[B] \rightarrow R_2^1[C] \doteq R_2^2[C]$, which satisfies the conditions (i)-(iii) for the choice $X = 1$, $Y = 2$: for (i) this set is $\{R[B]\}$; for (ii) the R^1-ES $\{R_1^1[B], R_2^1[B]\}$ satisfies the property; and for (iii) we use $L = \{R_1^1[A], R_1^2[A]\}$.

As mentioned in Section 2.2, for the given M and the query $\mathcal{Q}(x, z) : \exists y R(x, y, z)$, RQA is intractable [12]. This query is in UJCQ $\setminus$ CHAQ. Now, we have just obtained that RQA for that M is also intractable for *all* CHAQ queries. □

Example 12. Consider M consisting of $m_1 : \ R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$; $m_2 : \ R[A] \approx R[A] \wedge R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$. It does not satisfy the conditions of Theorem 2 (actually, Corollary 1). The sole L-component of m_1 is $\{R[A]\}$, and all attributes of this set occur in $LHS(m_2)$. M is easy, because the non-interacting set $\{R[A] \approx R[A] \rightarrow R[B] \doteq R[B], \ R[A] \approx R[A] \rightarrow R[C] \doteq R[C]\}$ is equivalent to it in the sense that, for any instance, the MRIs are the same for either set. This is because applying m_1 to the tuples of R and S results in an instance such that all pairs of tuples satisfying the first conjunct to the left of the arrow in m_2 satisfy the entire similarity condition. □

Theorem 2 gives a syntactic condition for hardness. It is an important result, because it applies to simple sets of MDs such as that in Example 5 that we expect to be commonly encountered in practice. Moreover, in Section 5, we use Theorem 2 to show that similar sets involving more than two MDs are also hard. The conditions for hardness in Theorem 2 are not necessary conditions. Actually, the set of MDs in Example 13 below is hard, but does not satisfy the conditions of this theorem.

4 A Dichotomy Result

All syntactic conditions/constructs on attributes above, in particular, the transitive closures on attributes, are "orthogonal" to semantic properties of the similarity relations. When similarity predicates are transitive, every linear pair not satisfying the hardness criteria of Theorem 2 is easy.

Theorem 3. Let (m_1, m_2) be a linear pair with $RHS(m_1) \cap RHS(m_2) = \emptyset$. If the similarity operators are transitive, then (m_1, m_2) is either easy or hard. More precisely, if the conditions of Theorem 2 hold, M is hard. Otherwise, M is easy. □

This result does not hold in general when similarity is not transitive (c.f. Proposition 2 below). The possibilities for accidental similarities are reduced by disallowing that two dissimilar values are similar to a same value. Actually, the complexity of the problem is reduced to the point where the resolved answer problem becomes tractable.

Example 13. The linear pair M consisting of $m_1: \ R[A] \approx S[B] \wedge R[I] \approx S[J] \rightarrow R[E] \doteq S[F]$; $m_2: \ R[E] \approx S[F] \wedge R[A] \approx S[J] \wedge R[I] \approx S[B] \rightarrow R[G] \doteq S[H]$ does not satisfy the conditions of Theorem 2, because m_1 has two L-components, $\{R[A], S[B]\}$ and $\{R[I], S[J]\}$. Since $LHS(m_2)$ includes one attribute of R and S from each of these L-components, conditions (a)(iii) and (b)(iii) are not satisfied. Then, by Theorem 3, M is easy when $\approx$ is transitive. □

Example 12 showed that a pair of MDs is easy for arbitrary $\approx$ by exhibiting an equivalent non-interacting set. This method cannot be applied in Example 13, because the similarity condition of m_1 is not included in that of m_2. The set of MDs in Example 13 can be hard for non-transitive similarity relations, as the following proposition shows.

Proposition 2. There exist (non-transitive) similarity operators $\approx$ for which the set of MDs in Example 13 is hard. □

5 Hardness of Acyclic Sets of MDs

We consider now acyclic sets of MDs of arbitrary finite size, concentrating on a class of them that is common in practice.

Definition 7. A set M of MDs is *pair-preserving* if for every attribute appearing in M, say $R[A]$, there is exactly one attribute appearing in M, say $S[B]$, such that $R[A] \approx S[B]$ or $R[A] \doteq S[B]$ (or the other way around) occurs in M. □

It is easy to verify that pair-preserving, acyclic sets of MDs are strongly acyclic. Pair-preservation typically holds in ER, because values for pairs of attributes are compared only if they hold the same kind of information (e.g. both addresses or both names).

Example 14. M in Example 12 is pair-preserving. The set of MDs in Example 13 is not pair-preserving, because $S[B]$ is paired with both $R[A]$ and $R[C]$ in m_1. It is also possible for cyclic sets of MDs to be pair-preserving. For example, the set $R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$; $R[B] \approx R[B] \rightarrow R[A] \doteq R[A]$ is pair-preserving. □

Now, recall from the previous section that syntactic conditions on linear pairs (m_1, m_2), like the absence of certain attributes in $LHS(m_1)$ from $LHS(m_2)$ (c.f. conditions (a)(iii) or (b)(iii)), imply hardness. *Non-inclusiveness* wrt. *subsets* of M is a syntactic condition on acyclic, pair-preserving sets M of MDs that generalizes those conditions that ensure hardness for linear pairs.

Definition 8. Let M be acyclic and pair-preserving, B an attribute in M, and $M' \subseteq M$. B is *non-inclusive* wrt. M' if, for every $m \in M \setminus M'$ with $B \in RHS(m)$, there is an attribute C such that: (a) $C \in LHS(m)$, (b) $C \notin \bigcup_{m' \in M'} LHS(m')$, and (c) C is *non-inclusive* wrt. M'. □

This is a recursive definition, with base case when C is not in $RHS(m)$ for any m (then is inclusive, i.e. not non-inclusive). Since $C \in LHS(m)$ in the definition, for any m_1 with $C \in RHS(m_1)$, there is an edge from m_1 to m. Therefore, we are traversing an edge backwards with each recursive step, and the recursion terminates by acyclicity.

Example 15. In the acyclic, pair-preserving set $\{m_1\colon\ R[I] \approx S[J] \rightarrow R[A] \doteq S[E];\ m_2\colon\ R[A] \approx S[E] \rightarrow R[C] \doteq S[B];\ m_3\colon\ R[G] \approx S[H] \rightarrow R[I] \doteq S[J]\}$, $R[A]$ is non-inclusive wrt. $\{m_2\}$ because $R[A] \in RHS(m_1)$ and there is an attribute, $R[I]$, in $LHS(m_1)$ that satisfies conditions (a), (b), and (c) of Definition 8. Conditions (a) and (b) are obviously satisfied. Condition (c) is satisfied, because $R[G]$ is non-inclusive wrt. $\{m_1\}$. This is trivially true, since $R[G] \notin RHS(m_1) \cup RHS(m_3)$. □

Non-inclusiveness is a generalization of conditions (a) (iii) and (b) (iii) in Theorem 2 to finite *sets* of MDs. It expresses a condition of inclusion of attributes in the LHS of one MD in the LHS of another. In particular, suppose $M = (m_1, m_2)$ is a pair-preserving linear pair, and take $M' = \{m_2\}$. It is easy to verify that the requirement that there is an attribute in $RHS(m_1)$ that is non-inclusive wrt. M' is equivalent to conditions (a)(iii) and (b)(iii) of Theorem 2. Theorem 4 tells us that a non-inclusive set of MDs is hard.

Theorem 4. Let M be acyclic and pair-preserving. Assume there is $\{m_1, m_2\} \subseteq M$, and attributes $C \in RHS(m_2)$, $B \in RHS(m_1) \cap LHS(m_2)$ with: (a) C is non-inclusive wrt $\{m_1, m_2\}$, and (b) B is non-inclusive wrt $\{m_2\}$. Then, M is hard. □

Example 16. (example 15 cont.) The set of MDs is hard. This follows from Theorem 4, with $\{m_1, m_2\}$ and C, B in Theorem 4 being $\{m_1, m_2\}$ and $R[C], R[A]$ in the example, resp. Part (b) of Theorem 4 was shown in the first part of this example. Part (a) holds trivially, since $R[C] \notin RHS(m_3)$. □

Example 17. Consider $M = \{m_1, m_2, m_3\}$ with $m_1\colon\ R[G] \approx S[H] \rightarrow R[I] \doteq S[J];$ $m_2\colon\ R[G] \approx S[H] \wedge R[I] \approx S[J] \rightarrow R[A] \doteq S[E];$ $m_3\colon\ R[G] \approx S[H] \wedge R[A] \approx S[E] \rightarrow R[C] \doteq S[B]$. It does not satisfy the condition of Theorem 4. The only candidates for $\{m_1, m_2\}$ in Theorem 4 are $\{m_1, m_2\}$ and $\{m_2, m_3\}$ in this example, because of the requirement that $RHS(m_1) \cap LHS(m_2) \neq \emptyset$. In the first case, B in the Theorem 4 is $R[I]$ (or $S[J]$), which does not satisfy (b) because $LHS(m_1) \backslash LHS(m_2) = \emptyset$. In the second case, B in Theorem 4 is $R[A]$ (or $S[E]$). Because $R[G]$ and $S[H]$ are in $LHS(m_3)$, $R[A]$ can only satisfy (b) if $R[I]$ does. $R[I]$ does not satisfy (b), since $LHS(m_1) \backslash LHS(m_3) = \emptyset$.

Actually, M is easy, because it is equivalent to the non-interacting set $m_1'\colon\ R[G] \approx S[H] \rightarrow R[I] \doteq S[J];\ m_2'\colon\ R[G] \approx S[H] \rightarrow R[A] \doteq S[E];\ m_3'\colon\ R[G] \approx S[H] \rightarrow R[C] \doteq S[B]$. This can be shown as in Example 12. □

Our dichotomy result applies to linear pairs (and transitive similarities). However, tractability can be obtained in some cases of larger sets of MDs for which hardness cannot be obtained via Theorem 4 (because the conditions do not hold). The following is a general result concerning sets such as M in Example 17.

Theorem 5. Let M be an acyclic, pair-preserving set of MDs. If, for each $m \in M$, all changeable attributes $A \in LHS(m)$ are inclusive wrt $\{m\}$, then M is easy. □

Example 18. (example 17 cont.) As expected, the set M of MDs $\{m_1, m_2, m_3\}$ satisfies the requirement of Theorem 5. To show this, the only attributes to be tested for inclusiveness wrt. an MD are $R[A]$ and $R[I]$. Specifically, it must be determined whether $R[I]$ is inclusive wrt $\{m_2\}$ and whether $R[A]$ is inclusive wrt $\{m_3\}$. $R[I]$ is inclusive wrt $\{m_2\}$, because all attributes in $LHS(m_1)$ are in $LHS(m_2)$. $R[A]$ is inclusive wrt $\{m_3\}$, since $R[G] \in LHS(m_3)$ and $R[I]$ is inclusive wrt $\{m_3\}$. □

Example 19. (example 16 cont.) $\{m_1, m_2, m_3\}$ in Example 15 was shown to be hard in Example 16. As expected, it does not satisfy the requirement of Theorem 5. This is because $R[A]$ is changeable, $R[A] \in LHS(m_2)$, and $R[B]$ is non-inclusive wrt $\{m_2\}$ since $R[I] \in LHS(m_1)$, $R[I] \notin LHS(m_2)$, and $R[I]$ is non-inclusive wrt $\{m_2\}$. □

The conditions of Theorems 4 and 5 are mutually exclusive: B in Theorem 4 is changeable (since $B \in RHS(m_1)$), $B \in LHS(m_2)$, and B is non-inclusive wrt $\{m_2\}$. Together, they do not provide a dichotomy result, as the following example shows.

Example 20. The set formed by $m_1: \ R[E] \approx R[E] \rightarrow R[B] \doteq R[B]$; $m_2: \ R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$; $m_3: \ R[E] \approx R[E] \rightarrow R[C] \doteq R[C]$ does not satisfy the condition of Theorem 5, because $R[B]$ is changeable and non-inclusive wrt. $\{m_2\}$. Nor condition (a) of Theorem 4, because C is inclusive wrt. $\{m_1, m_2\}$ ($R[E] \in LHS(m_1)$).

The tractability of this case cannot be determined through the theorems above, but it is easy, because, for any update sequence that leads to an MRI, each set of merged duplicates must be updated to a value in the set (to satisfy minimality of change). It is easily verified that, with this restriction, the second update to the values of $R[C]$ is subsumed by the first, and therefore this update has no effect on the instance. Thus, sets of duplicates can be computed in the same way as with non-interacting sets. □

Notice that the condition of Theorem 2 that there exists an ES that is not bounded does not appear in Theorem 4. This is because, for pair-preserving, acyclic sets of MDs, this condition is always satisfied by any subset of the set that is a linear pair. Indeed, for such a subset (m_1, m_2), if all its ESs are bounded, then by the pair-preserving requirement, $LHS(m_2) \subseteq LHS(m_1)$. Since (m_1, m_2) is a linear pair, $LHS(m_2) \cap RHS(m_1) \neq \emptyset$. This implies $LHS(m_1) \cap RHS(m_1) \neq \emptyset$, contradicting the acyclicity assumption.

For linear pairs, Theorem 4 becomes Theorem 2. For such pairs, condition (a) of Theorem 4 is always satisfied. If the (acyclic) linear pair is also a pair-preserving, as required by Theorem 4, the conditions of Theorem 2 reduce to conditions (a)(iii) and (b)(iii), which, as noted previously, are equivalent to condition (b) of Theorem 4.

6 Conclusions

We have shown that RQA is typically intractable when the MDs have non-cyclic dependencies on each. Our results depend on our chase-based semantics. Alternatives to this chase have been considered [6]. A quite different chase, which applies one MD at a time and uses *matching functions*, is presented in [5, 2].

The definition of resolved answer reminds us of *consistent query answering* (CQA) [4], where much research has been about (polynomial-time) query rewriting methodologies. In all the cases identified in the literature (see [4, 18] for recent surveys), the rewritings have been first-order. For MDs, the exhibited rewritings are in Datalog [13].

RQA brings many new challenges in comparison to CQA, and results for the latter cannot be applied (at least not in an obvious manner): (a) MDs contain usually non-transitive similarity relations. (b) Enforcing consistency of updates requires computing the transitive closure of such relations. (c) *Tuple-based repairs* are usually considered in CQA [4]. The minimality of *value changes*, not always used in CQA, has not been considered for consistent rewritings. (d) The semantics of resolved query answering for MD-based ER is given, in the end, in terms of a chase procedure.[3] However, the se-

[3] See [17] for some implicit connections between repairs and chase procedures, e.g. as used in data exchange; and [8] for connections with the chase used for database completion with ICs.

mantics of CQA is model-theoretic, given in terms of non-operationally defined repairs that arise from set-theoretic conditions. For additional discussions of differences and connections between CQA and RQA, see [12, 14].

We have presented the first dichotomy result for the complexity of RQA. Its cases depend on the set of MDs, for a fixed class of queries. In CQA with FDs, dichotomy results have been obtained for limited classes of conjunctive queries [15, 16, 18]. However, in CQA the cases depend mainly on the queries, as opposed to FDs.

Some open problems for ongoing and future research are: (a) Extending the class of CHAQ queries, considering additional projections, and also boolean queries. (b) Deriving a dichotomy result for acyclic, pair-preserving sets analogous to the one for linear pairs. (c) Since, FDs (and other equality generating dependencies) can be expressed as MDs, with equality as a transitive symmetry relation, applying the dichotomy result in Theorem 3 to CQA under FDs under a *value-based* repair semantics [4].

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Z. Bahmani, L. Bertossi, S. Kolahi and L. Lakshmanan. Declarative entity resolution via matching dependencies and answer set programs. Proc. KR 2012.

[3] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. Euijong Whang, and J. Widom. Swoosh: A generic approach to entity resolution. *VLDB Journal*, 2009, 18(1):255-276.

[4] L. Bertossi. *Database Repairing and Consistent Query Answering*, Morgan & Claypool, Synthesis Lectures on Data Management, 2011.

[5] L. Bertossi, S. Kolahi and L. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *Theory of Computing Systems*, 2013, 52(3):441-482.

[6] L. Bertossi and J. Gardezi. Tractable vs. Intractable Cases of Matching Dependencies for Query Answering under Entity Resolution. Corr ArXiv: 1309.1884, 2013.

[7] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 2008, 41(1):1-41.

[8] A. Cali, D. Lembo and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. Proc. PODS 2003, pp. 260-271.

[9] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowledge and Data Eng.*, 2007, 19(1):1-16.

[10] W. Fan. Dependencies revisited for improving data quality. Proc. PODS 2008.

[11] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. Proc. VLDB 2009.

[12] J. Gardezi, L. Bertossi, and I. Kiringa. Matching dependencies: semantics, query answering and integrity constraints. *Frontiers of Computer Science*, Springer, 2012, 6(3):278-292.

[13] J. Gardezi, L. Bertossi. Query rewriting using datalog for duplicate resolution. Proc. Datalog 2.0, 2012, Springer LNCS 7494, pp. 86-98, 2012.

[14] J. Gardezi and L. Bertossi. Tractable cases of clean query answering under entity resolution via matching dependencies. Proc. SUM 2012, Springer LNAI 7520, pp. 180-193, 2012.

[15] P. Kolaitis and E. Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Information Processesing Letters*, 2012, 112(3):7785.

[16] P. Koutris and D. Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. Proc. ICDT 2014, pp. 165-176.

[17] B. ten Cate, G. Fontaine and P. Kolaitis. On the data complexity of consistent query answering. Proc. ICDT 2012, pp. 22-33.

[18] J. Wijsen. A survey of the data complexity of consistent query answering under key constraints. Proc. FoIKS 2014, Springer LNCS 8367, 2014, pp. 62-78.