Carleton UNIVERSITY

Matching Dependencies for Entity Resolution in Databases

Leopoldo Bertossi
Carleton University
Ottawa, Canada

Joint work with:

Jaffer Gardezi

University of Ottawa

Canada

Duplicate Resolution and MDs

A database may contain several representations of the same external entity

The database contains "duplicates", usually considered to be undesirable

The database has to be cleaned ...

The problem of duplicate- or entity-resolution is about:

(a) detecting duplicates, and

(b) merging duplicate representations into single representations

This is a classic and complex problem in data management, and data cleaning in particular

We concentrate on merging, in a relational context

A generic way to approach the problem consists in specifying what attribute values have to be matched (made identical) under what conditions

A declarative language with a precise semantics could be used for this purpose

Matching Dependencies (MDs) were proposed

(Fan et al., PODS'08, VLDB'09)

They are rules for resolving pairs of duplicate representations (two tuples at a time)

Example: The similarities of phone and address indicate that the tuples refer to the same person, and the names should be matched

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

Here: 723-9583 $\approx$ (750) 723-9583 and 10-43 Oak St. $\approx$ 43 Oak St. Ap. 10

An MD capturing this cleaning policy:

$$P[Phone] \approx P[Phone] \wedge P[Address] \approx P[Address] \ \rightarrow \ P[Name] \doteq P[Name]$$

(an MD may involve two different relations)

Dynamic interpretation: The values on the RHS should be updated to some (unspecified) common value

Matching Dependencies

MDs are rules m of the form

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

The left-hand side captures a similarity condition on pairs of tuples, in relations R and S

Abbreviation: $m\colon \ R[\bar{A}] \approx S[\bar{B}] \ \rightarrow \ R[\bar{C}] \doteq S[\bar{E}]$

$LHS(m)$: set of attributes on the left-hand side of of the arrow

$RHS(m)$: similarly

Attributes in $RHS(m)$, for some m: *changeable attributes*

The similarity operators $\approx$ satisfy:

(a) Symmetry: If $x \approx y$, then $y \approx x$

(b) Equality Subsumption: If $x = y$, then $x \approx y$

Transitivity *not* always assumed (and may not hold)

MDs are to be “applied” iteratively until duplicates are solved

To keep track of changes and comparing tuples and instances, we use global tuple identifiers, a non-changeable surrogate key

Usually shown as: $R(t, \bar{x})$

A *position*: A pair (t, A) with t a tuple id, A an attribute

The *position’s value* is $t[A]$, the value for A in tuple (with id) t

MD Semantics

A semantics for MDs acting on database instances was introduced in (Gardezi and Bertossi; LID'11; FofCS'12)

Based on a chase procedure starting with original instance D

A *resolved instance* D' is obtained from a finitely terminating sequence of instances $D \mapsto D_1 \mapsto D_2 \mapsto \cdots \mapsto D'$

D' satisfies the MDs in the sense of the static interpretation, seeing MDs as EGDs

The semantics specifies the one-step transitions ($\mapsto$) or updates allowed to go from D_{i-1} to D_i

For a given instance, only values of *modifiable positions* allowed to change in such a step, and as forced by the MDs

(syntactically and recursively depend on M and current instance)

Example:

$$R[A] = R[A] \;\rightarrow\; R[B] \doteq R[B]$$
$$R[B] = R[B] \;\rightarrow\; R[C] \doteq R[C]$$

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

Attribute $R(C)$ is changeable

Position (t_2, C) is not modifiable wrt M and D: No justification to change its value in one step on the basis of an MD and D

Position (t_1, C) is modifiable

Two resolved instances for D: D_1 and D_2

$R(D_1)$	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

D_1 cannot be obtained in a single (one step) update: the red value is for a non-modifiable position

D_2 can ...

Single-Step Semantics:

Each pair D_i, D_{i+1} in an update sequence (a chase step) must *satisfy* the set M of MDs relative to modifiable attributes, denoted $(D_i, D_{i+1}) \models_{ma} M$

$(D_i, D_{i+1}) \models_{ma} M$ holds iff:

1. For every MD, 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}
2. The value of a position can only differ between D_i and D_{i+1} if it is modifiable wrt D_i

Notice: A *resolved instance* D' is *stable* in the sense that

$$(D', D') \models_{ma} M$$

This semantics stays as close as possible to the spirit of the MDs as originally introduced, and also uncommitted wrt values for matchings

Other semantics have been proposed (are being) and investigated

- As above, but modifying the chase conditions, e.g.

  - One MD at a time

  - Previous resolutions cannot be unresolved

- Using matching functions to choose a value for a match

  (Bertossi, Kolahi, Lakshmanan; ICDT'11, TofCSs 2013),

  (Bahmani, Bertossi, Kolahi, Lakshmanan; KR'12)

Minimally Resolved Instances:

Among the resolved instances we prefer those that are closest to the original instance

A minimally resolved instance (MRI) of D is a resolved instance D' such that the number of changes of attribute values comparing D with D' is a minimum

Instance D_2 in Example 2 is an MRI, but not D_1 (2 vs. 3)

Resolved Answers:

Given a conjunctive query $\mathcal{Q}$, a set of MDs M, and an instance D, the *resolved answers* are invariant under the entity resolution process

That is, the *resolved answers* are those answers to $\mathcal{Q}$ that are true in all MRIs of D

The RQA decision problem:

$RA(\mathcal{Q}, M) = \{\ D, \bar{a} \mid \bar{a}$ is resolved answer to $\mathcal{Q}$ from D wrt $M\}$

Resolved query answering (RQA) in the spirit of consistent query answering (CQA) from an instance that fails to satisfy a set of integrity constraints (Arenas, Bertossi, Chomicki; PODS'99)

Developing (polynomial-time) query rewriting methodologies for (conjunctive) CQA has been the focus of intensive research

Such rewritings, in cases when conjunctive CQA is polynomial-time, have been first-order

Doing something similar for MDs has not been attempted before and brings new challenges:

- MDs contain the non-transitive similarity predicates
- Enforcing consistency of updates requires computing the transitive closure of such operators

  Example: $R[A] \approx R[A] \ \rightarrow \ R[B] \doteq R[B]$

$R(D)$	A	B
t_1	a	e
t_2	b	f
t_3	c	g

It holds $a \approx b \approx c$, $a \not\approx c$

Consistently updating requires $\approx$'s transitive closure

Duplicate resolution requires $t_1[B]$ and $t_3[B]$ to be updated to the same value

- Minimality of value changes (as opposed to tuple changes as usual in CQA) (but see below)

(In)Tractability of RQA?

Deciding resolved query answers is generally intractable

E.g., it is intractable for the query $\mathcal{Q}(x,z)\!:\ \exists y R(x,y,z)$ and MDs

$$\begin{array}{rcl} m_1 \ : R[A] \approx R[A] & \rightarrow & R[B] \doteq R[B] \\ m_2 \ : R[B] \approx R[B] & \rightarrow & R[C] \doteq R[C] \end{array}$$

The MDs here do not depend “cyclically” on each other ...

The query is very simple (Gardezi,Bertossi; LID’11)

There sets of MDs for which RQA is tractable, for a broad class of conjunctive queries

Distinction between cyclic and acyclic cases is crucial

Efficient Query Answering/Rewriting?

We developed a query rewriting methodology

The rewritten queries turn out to be Datalog queries

There are two main classes of sets of MDs that enjoy query rewriting:

(Gardezi, Bertossi; SUM'12, Datalog 2.0'12)

1. Sets where MDs do not depend on each other: *non-interacting* sets of MDs

2. Some sets where MDs cyclically depend on each other

E.g.

$$R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$R[B] \approx R[B] \rightarrow R[A] \doteq R[A]$$

But not

$$R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$$

(as seen, intractable for simple queries)

Cyclic Cases of Sets of MDs

Example:

$R(D)$	A	B
1	a_1	d_1
2	a_2	e_2
3	b_1	e_1
4	b_2	d_2

$$m_1: \; R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$m_2: \; R[B] \approx R[B] \rightarrow R[A] \doteq R[A]$$

(with equality as similarity)

A possible update sequence:

$R(D)$	A	B
1	a_1	d_1
2	a_2	e_2
3	b_1	e_1
4	b_2	d_2

$\mapsto$

$R(D_1)$	A	B
1	b_2	d_1
2	a_2	d_1
3	a_2	e_1
4	b_2	e_1

$\mapsto$

$R(D_2)$	A	B
1	a_2	e_1
2	a_2	d_1
3	b_2	d_1
4	b_2	e_1

$\mapsto$

Instances exhibit an alternating behaviour, which can only terminate by updating all values in the A and B columns to a (most frequent) common value

MRIs have a simple form

HSC Sets of MDs: (will enjoy query rewritability)

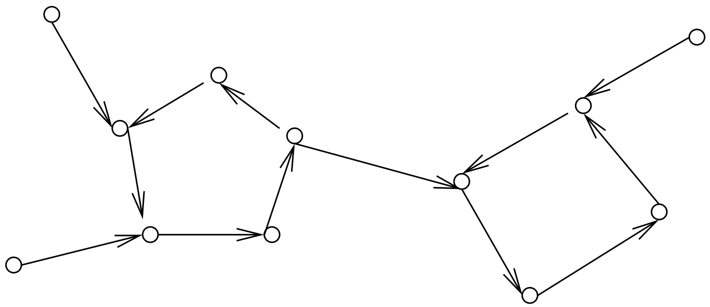
More generally, MRIs take a simple and easily characterizable form for hit-simple-cyclic (HSC) sets of MDs

For M set of MDs, its directed graph $MDG(M)$:

- Each MD $m \in M$ as a vertex
- An edge from m_1 to m_2 if there is an attribute on RHS of m_1 that is on LHS of m_2

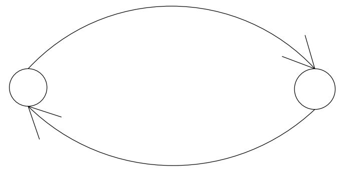
M in an HSC set when:

- For each $m \in M$, at most one attribute in $LHS(m)$ is changeable (i.e. appears in some RHS in M)
- Each vertex m_1 in $MDG(M)$ is on at least one cycle, or there is an edge from m_1 to a vertex m_2 that is on a cycle



Example: (a) An HSC set of MDs

$$m_1: \; R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$m_2: \; R[B] \approx R[B] \rightarrow R[A] \doteq R[A]$$



(b) A non-HSC set of MDs

$$m_1: \; R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$m_2: \; R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$$

$m_1 \circ \longrightarrow \circ\, m_2$

(c) A non-interacting set of MDs

$$m_1: \; R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$m_2: \; R[C] \approx R[C] \rightarrow R[D] \doteq R[D]$$

$m_1 \circ \quad \circ\, m_2$

Example: Relation R about people

Attributes A and C for address and email

Tuples with very similar addresses or identical emails likely refer to the same person

Two “key” MDs

$$m_1\colon\ R[A] \approx R[A] \rightarrow R[C,F,G] \doteq R[C,F,G],$$
$$m_2\colon\ R[C] \approx R[C] \rightarrow R[A,F,G] \doteq R[A,F,G].$$

A common situation ...

An HSC set that enjoys query rewriting

Here cycles help us, because the termination condition for the chase imposes a relatively simple form on the minimally resolved instances (easier to capture and characterize)

We haven't mentioned queries yet ...

For HSC and non-interacting sets of MDs, conjunctive queries in a class can be rewritten to retrieve the resolved answers

Queries with no joins on existentially quantified join variables corresponding to changeable attributes: unchangeable-join-attribute conjunctive (UJCQ) queries (G&B; SUM'12)

Example: Schema $R[A, B, C]$

MD $R[A] = R[A] \rightarrow R[B, C] \doteq R[B, C]$

$\mathcal{Q}\colon\ \exists x \exists y \exists z (R(x, y, c) \wedge R(z, y, d))$ is not UJCQ

$\mathcal{Q}'\colon\ \exists x \exists z (R(x, y, z) \wedge R(x, y', z'))$ is UJCQ

Outside UJCQ, RQA tends to be intractable, with or without cycles in MDs ...

Tuple-Attribute Closure:

Given instance D and MDs M

$TA_{M,D}$, the tuple-attribute closure, is the transitive closure of a binary relation $\approx'$ on positions

$\approx'$ depends on D, M, and $\approx$ (example below)

$\approx'$ and its TC can be defined in Datalog

$TA_{M,D}$ turns out to be an equivalence relation

For an equivalence class E of $TA_{M,D}$:

$$freq^D(a, E) := |\ \{(t, A) \mid (t, A) \in E,\ t[A] = a \text{ in } D\}\ |$$

In an MRI, all positions in each equivalence class E must take a common value a that maximizes $freq^D(a, E)$

Example: $M = \{R[A] \approx R[A] \rightarrow R[B] \doteq R[B]\}$

It holds $a \approx b \approx c$

$R(D)$	A	B
t_1	a	e
t_2	b	e
t_3	c	g

$\rightarrow$

$R(D')$	A	B
t_1	a	e
t_2	b	e
t_3	c	e

$TA_{M,D}$ is TC of the relation $(t_1, B) \approx' (t_2, B) \approx' (t_3, B)$

D' is the only MRI

The Rewriting:

Input: a conjunctive query

Output: a stratified Datalog^{not} program with recursion and aggregation (no disjunction)

Recursion arises in the computation of the TC $TA_{M,D}$

Aggregation is needed to compute $freq^D(a, E)$

Negation needed to maximize the frequency (or minimize value changes)

Notation for aggregation: $P(\bar{x}, count(\bar{u})) \leftarrow Body(\bar{y})$

(count distinct with group-by, $\bar{x} \cup \bar{u} \subseteq \bar{y}$)

To retrieve the resolved answers to $\mathcal{Q}$ from D:

1. Generate a query rule defining auxiliary query predicate $\mathcal{Q}'$ from $\mathcal{Q}$
2. Combine with the Datalog program above
3. Run on top of D

Resolved answers can be obtained in polynomial time (in data)

Query Rewriting Example

Example: $R[A,B]$ Query: $\mathcal{Q}(x,y) \leftarrow R(x,y)$

M:

$$R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$R[B] \approx R[B] \rightarrow R[A] \doteq R[A]$$

We use tuple identifiers, so $R(x,y)$ becomes $R(t,x,y)$

$$(t_1, A) \approx' (t_2, A) \leftarrow R(t_1, \bar{x}), R(t_2, \bar{y}), t_1[A] \approx t_2[A]$$
$$(t_1, A) \approx' (t_2, A) \leftarrow R(t_1, \bar{x}), R(t_2, \bar{y}), t_1[B] \approx t_2[B]$$
$$(t_1, B) \approx' (t_2, B) \leftarrow R(t_1, \bar{x}), R(t_2, \bar{y}), t_1[A] \approx t_2[A]$$
$$(t_1, B) \approx' (t_2, B) \leftarrow R(t_1, \bar{x}), R(t_2, \bar{y}), t_1[B] \approx t_2[B]$$

$\approx'$ relates both attribute A and B positions of pairs of tuples satisfying the similarity condition of either MD

Now the transitive closure:

Reflexivity:

$$TA(t, A, t, A) \leftarrow R(t, \bar{x})$$
$$TA(t, B, t, B) \leftarrow R(t, \bar{x})$$

Symmetry:

$$TA(t_1, A, t_2, A) \leftarrow TA(t_2, A, t_1, A)$$
$$TA(t_1, B, t_2, B) \leftarrow TA(t_2, B, t_1, B)$$

Transitivity:

$$TA(t_1, A, t_2, A) \leftarrow (t_1, A) \approx' (t_2, A)$$
$$TA(t_1, B, t_2, B) \leftarrow (t_1, B) \approx' (t_2, B)$$
$$TA(t_1, A, t_3, A) \leftarrow TA(t_1, A, t_2, A), (t_2, A) \approx' (t_3, A)$$
$$TA(t_1, B, t_3, B) \leftarrow TA(t_1, B, t_2, B), (t_2, B) \approx' (t_3, B)$$

Frequencies:

$$C^A(t_1, u, count(t_2)) \leftarrow TA(t_1, A, t_2, A), R(t_1, x, y), R(t_2, u, v)$$
$$C^B(t_1, v, count(t_2)) \leftarrow TA(t_1, B, t_2, B), R(t_1, x, y), R(t_2, u, v)$$

In $C^A(t_1, u, count(t_2))$, the value of the count expression is $freq(u, E)$, where E is the equivalence class of TA to which (t_1, A) belongs

Minimizing:

$$Compare^A(t, x) \leftarrow C^A(t, x, z_1), C^A(t, x', z_2), z_1 \leq z_2, x \neq x'$$
$$Compare^B(t, y) \leftarrow C^B(t, y, z_1), C^B(t, y', z_2), z_1 \leq z_2, y \neq y'$$

$Compare^A(t, x)$ is true iff there is a value x' whose frequency in the equivalence class of TA closure to which (t, A) belongs is at least as large as that of x

The query rule:

Resolved answers to original query

$$\mathcal{Q}(x,y) \leftarrow R(x,y)$$

are the answers to rewritten query (with answer predicate) $\mathcal{Q}'$:

$$\begin{aligned}\mathcal{Q}'(x,y) \;\; \leftarrow \;\; & R(t,x,y), C^A(t,x,z_1), C^B(t,y,z_2), \\ & not\ Compare^A(t,x), not\ Compare^B(t,y),\end{aligned}$$

The Repairs and CQA Connection

We consider MDs with equality for similarity, and key constraints on the CQA side

In some cases, computing the resolved answers can be reduced in PTIME to consistent query answering, allowing us to take advantage of results from CQA

In CQA, the repairs of an instance of a relation R that fails to satisfy a key constraint $R\colon A \rightarrow B$ are obtained by restoring consistency with the minimal number of tuple deletions

Instance

$R(D)$	A	B
	a	b
	a	c

Repairs

$R(D)$	A	B
	a	b

$R(D)$	A	B
	a	c

Since duplicate tuples are discarded, the repairs coincide with the MRIs for the MD $R[A] = R[A] \rightarrow R[B] \doteq R[B]$

Resolved Answers from Consistent Answers:

Consider an instance D of a relation $R[\bar{A}, \bar{B}]$ with MD

$$R[\bar{A}] = R[\bar{A}] \rightarrow R[\bar{B}] \doteq R[\bar{B}]$$

There is a transformation T such that the resolved answers to any query $\mathcal{Q}$ for $R(D)$ are the consistent answers to $\mathcal{Q}$ wrt the FD $R\colon \bar{A} \rightarrow \bar{B}$ on instance $T(R(D))$

T is a first-order query with aggregation (counting)

First-order query rewriting can be used to find the consistent answers wrt key constraints for important classes of conjunctive queries (Fuxman and Miller; ICDT'05) (Wijsen; PODS'10)

If $\mathcal{Q}'$ is the rewriting of $\mathcal{Q}$, then $\mathcal{Q}' \circ T$ will return the resolved answers to $\mathcal{Q}$ when posed to the original instance D

Example: Relation $R[A, B, C]$ has key constraint $R\colon A \rightarrow BC$

$R(D)$	A	B	C
	a	c	e
	a	d	e
	b	c	e
	b	c	f
	b	d	f
	b	d	h
	b	g	h

$$\xrightarrow{T}$$

$T(R(D))$	A	B	C
	a	c	e
	a	d	e
	b	c	f
	b	c	h
	b	d	f
	b	d	h

For each value of the key, the values of the other attributes are obtained by taking the “cross product” of the most frequently-occurring values for those attributes

Example: For the query $\mathcal{Q}:\ \exists x\exists y\exists z\exists w(R(x,y,w) \wedge S(y,w,z))$ with relational predicates $R[A,B,C]$ and $S[C,E,F]$ and KCs $R\colon A \rightarrow BC$ and $R\colon CE \rightarrow F$,

$\mathcal{Q}'$ retrieves the consistent answers (Fuxman and Miller; ICDT'05)

$$\mathcal{Q}'\colon\ \exists x\exists y\exists z\exists w[R(x,y,w) \wedge S(y,w,z) \wedge \forall y'\forall w'(R(x,y',w') \rightarrow \exists z' S(y',w',z'))$$

Transformation T maps R to R'

$$R'(x,y,w) := \exists w'\{R(x,y,w') \wedge \forall y'[Count\{w'' \mid R(x,y',w'')\} \leq Count\{w'' \mid R(x,y,w'')\}]\} \wedge Count\{w'' \mid R(x,y,w'')\}]\} \wedge \exists y'\{R(x,y',w) \wedge \forall v[Count\{y'' \mid R(x,y'',v)\} \leq Count\{y'' \mid R(x,y'',w)\}]\}$$

The resolved answers for the corresponding key MDs are obtained from $\mathcal{Q}' \circ T$ (replace R by R' in $\mathcal{Q}$)

An Intractability Result via CQA:

Consider the relational predicate $R[A,B,C]$, the MD

$$m\colon\ R[A] = R[A] \rightarrow R[B,C] \doteq R[B,C]$$

and the query

$$\mathcal{Q}\colon \exists x \exists y \exists y' \exists z (R(x,y,c) \wedge R(z,y',d) \wedge y = y')$$

Deciding RQA is $coNP$-complete (in data)

This follows from a corresponding result for CQA with the FD corresponding to the MD (the repairs are MRIs for the particular instance produced in the reduction) (Chomicki et al.; IandC'04)

MD is non-interacting, and the query is not UCJQ

This shows that the tractability result for UCJQ queries cannot be extended to all conjunctive queries

Some Intractability Results

Above we have seen mainly tractable cases of RQA

(Gardezi, Bertossi; SUM'12, Datalog 2.0'12)

But also an intractable case: For

$$R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$$

RQA is intractable for a simple UJCQ query: $\mathcal{Q}(x, z)$: $\exists y R(x, y, z)$

More general results concerning interacting, acyclic MDs?

[Corr arXiv:1309.1884v1, 2013]

In the absence of cycles, RQA tends to be intractable

We investigate RQA for a subclass of UJCQ: changeable attribute queries (CHAQ)

We still concentrate on UJCQ: Outside RQA can be intractable even for single MDs; inside we find tractable and hard cases

We focus mainly on the kinds of interaction of MDS than in most general classes of queries for which (in)tractability holds

This in contrast with CQA, where interaction of FDs (under tuple deletion repairs) is less of an issue and mostly queries determine (in)tractability

Set M of MDs involving predicates $R, S, \ldots$

$\mathcal{Q}$ a UJCQ

$\mathcal{Q}$ is changeable attribute query (CHAQ) when it contains a conjunct of the form $R(\bar{x})$ with all variables free (a "free occurrence" of R) (a source of intractability)

Example: (with M as above)

- $\mathcal{Q}(x)\colon\ \exists y \exists z R(x, y, z)$ is not CHAQ

  Actually, RQA is trivially tractable if x corresponds to an unchangeable attribute (RAs are the usual answers)

- $\mathcal{Q}'(x, y, z)\colon\ \exists w \exists t (R(x, y, z) \wedge S(x, w, t))$ is CHAQ

Some Terminology and Notation: M set of MDs

- M is hard: For every CHAQ $\mathcal{Q}$, $RA(\mathcal{Q}, M)$ is *NP*-hard

  M is easy: For every CHAQ $\mathcal{Q}$, $RA(\mathcal{Q}, M)$ is in *PTIME*

  Of course, M does not have to be hard or easy

- M is acyclic if $MDG(M)$ is acyclic

Most of results hold for pairs of MDs, we concentrate on this case first, say $M = \{m_1, m_2\}$

Intractability Criteria for RQA

We first consider linear pairs of MDs $M = (m_1, m_2)$: acyclic, on two distinct relational predicates, an edge from m_1 to m_2

For $m \in M$:

- $LRel(m)$: Symmetric binary relation, relates attributes A, B where $R[A] \approx S[B]$ appears in $LHS(m)$
- $RRel(m)$: Similarly, with $R[A] \doteq S[B]$, and $RHS(m)$
- L-component of m: Equivalence class of the reflexive and transitive closure of $LRel(m)$
- R-component: Similarly, with $RRel(m)$

Example: $m\colon\ R[A] \approx S[B] \wedge R[A] \approx S[C] \ \rightarrow\ R[E] \doteq S[F] \wedge R[G] \doteq S[H]$

Only one L-component: $\{R[A], S[B], S[C]\}$

Two R-components: $\{R[E], S[F]\}$ and $\{R[G], S[H]\}$

Equivalence Sets of Attributes: (m_1, m_2) a linear pair with relational predicates R, S

- B_R: Reflexive, symmetric, binary relation on $Attr(R)$
  $(R[U_1], R[U_2]) \in B_R$ iff $R[U_1], R[U_2]$ are in same R-component of m_1 or same L-component of m_2 (similarly B_S)
- R-equivalent set (R-ES): Equivalence class of $TC(B_R)$, with at least one attribute in $LHS(m_2)$ (similarly S-ES)
- An (R or S)-ES E is bounded if $E \cap LHS(m_1) \neq \emptyset$
  (otherwise, unbounded)

Example: Schema $R[A, C, F, H, I, M]$, $S[B, D, E, G, N]$

Linear pair (m_1, m_2):

$$m_1 : \; R[A] \approx S[B] \rightarrow R[C] \doteq S[D] \; \wedge R[C] \doteq S[E] \wedge \\ R[F] \doteq S[G] \wedge R[H] \doteq S[G]$$

$$m_2 : \; R[F] \approx S[E] \wedge R[I] \approx S[E] \; \wedge \\ R[A] \approx S[E] \wedge R[F] \approx S[B] \quad \rightarrow \quad R[M] \doteq S[N]$$

- $B_R(R[F], R[H])$ because of $R[F] \doteq S[G]$, $R[H] \doteq S[G]$
- $B_R(R[F], R[I])$ because of $R[F] \approx S[E]$, $R[I] \approx S[E]$
- $B_R(R[I], R[A])$ because of $R[I] \approx S[E]$, $R[A] \approx S[E]$
- $\{R[A], R[F], R[I], R[H]\}$ is an R-ES, and bounded due to $\{R[A], R[F], R[I], R[H]\} \cap LHS(m_1) = \{R[A]\} \neq \emptyset$

Theorem 1: (m_1, m_2) linear pair, with relational predicates R and S, E_R, E_S be the sets of R-ESs and S-ESs, resp.

(m_1, m_2) is hard if $RHS(m_1) \cap RHS(m_2) = \emptyset$, and at least one of (a) and (b) 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

Example: The linear pair (m_1, m_2) is hard:

$$m_1 : \; R[A] \approx S[B] \rightarrow R[C] \doteq S[D]$$
$$m_2 : \; R[C] \approx S[D] \rightarrow R[E] \doteq S[F]$$

First: $RHS(m_1) \cap RHS(m_2) = \emptyset$

It satisfies condition (a):

Condition (a)(i) holds: $R[C] \in RHS(m_1) \cap LHS(m_2)$

Conditions (a)(ii) and (a)(iii) trivially satisfied:

There are no attributes of $LHS(m_1)$ in $LHS(m_2)$

A Dichotomy Result

All syntactic conditions/constructs on attributes above are "orthogonal" to semantic properties of similarity

In particular, for all the transitive closures on attributes above

When similarity predicates are transitive, every linear pair not satisfying the hardness criteria of Theorem 1 is easy

Theorem 2: Let (m_1, m_2) be a linear pair with $RHS(m_1) \cap RHS(m_2) = \emptyset$

If the similarity predicates are transitive, then (m_1, m_2) is either easy or hard

Intractability Criteria beyond Acyclic Pairs

We consider finite sets of acyclic MDs of arbitrary size

The generalization from acyclic pairs of MDs is based on the notions of pair-preserving acyclic and non-inclusive sets of MDs

A set M of MDs is pair-preserving if, for every attribute $R[A]$ occurring in M, there is only one attribute $S[B]$ with $R[A] \approx S[B]$ or $R[A] \doteq S[B]$ occurring in MDs in M

This condition typically holds in entity resolution: Values of pairs of attributes are normally compared only if they hold the same kind of information (e.g. both addresses or both names)

Syntactic conditions on pairs (m_1, m_2) imply hardness (Th1)

One of its requirements is the absence of certain attributes in $LHS(m_1)$ from $LHS(m_2)$ (conditions (a)(iii) or (b)(iii))

Non-inclusiveness wrt subsets of M is a syntactic generalization that ensures hardness for acyclic, pair-preserving Ms

Theorem 3: M acyclic and pair-preserving

If 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

As expected, Theorem 3 reduces to Theorem 1 for pair-preserving linear pairs

Final Remarks

- We have shown that resolved query answering is typically intractable when the MDs have a non-cyclic dependence on each other

- Other definitions of resolved answer can be considered in our setting, such as

  (a) Answers that are true in *all* (not necessarily minimal) resolved instances

  (b) Answers in all (minimal) resolved instances obtained with a *modified* chase procedure that never makes unequal values that have been made equal before

  - The same rewriting techniques apply, but now to some sets of MDs with non-cyclic dependencies

  - For acyclic pairs of MDs, we may obtain different behaviors wrt the semantics in this work

Example:

$$R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$$
$$R[B] \approx R[B] \rightarrow R[C] \doteq R[C]$$

RQA is tractable for every UJCQ under this alternative semantics

- Many open problems with the semantics used in this work, and the alternative ones ...

- Some of them include the relationship between repairs/CQA and MRI/RQA

Results for/from CQA for value-based repairs?

- Implementation of the query rewriting approaches

Ongoing work ...