Some Declarative Approaches to Data Quality

Leopoldo Bertossi
Carleton University
Ottawa, Canada
bertossi@scs.carleton.ca

BICOD, Birkbeck College, London, 2017

New Challenges from Data Quality Research

- Data quality has many dimensions: consistency, completeness, accuracy, redundancy, freshness, ...

  All of them create in the end uncertainty in data

- Data quality assessment (DQ) and data cleaning (DC) have been mostly: Ad-hoc, rigid, vertical, and application-dependent activities

- There is a lack of fundamental research in data quality assessment and cleaning

- Things are starting to change ...

- Recently, different forms of data quality constraints have been proposed and investigated
- They provide generic languages for expressing quality concerns

  Suitable for specifying adaptive and generic data quality assessment and data cleaning techniques

Contents:

A. Inconsistent Data

B. Matching Dependencies and Entity Resolution

C.1. Contexts in Data Quality

C.2. Ontological Multidimensional Contexts and Data Quality

This is not a survey, but an introduction to some research topics I have been involved in

A. Inconsistent Data

Inconsistent Databases

- Consistency has to do with satisfying semantic constraints, usually in the form of integrity constraints (ICs)

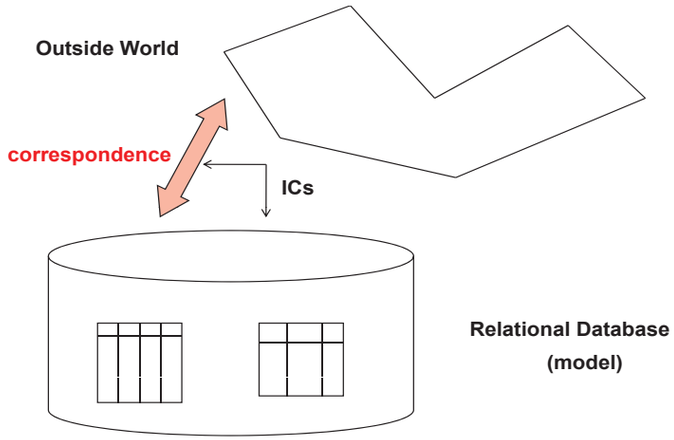
  ICs have been around for a long time ...

  They are used to capture the application semantics in the data model and database

  They have been studied in general and have wide application in data management

  Much fundamental/technical research has been developed

- A database instance D is a model of an outside reality



- An integrity constraint on D is a condition that D is expected to satisfy in order to capture the semantics of the application domain
- A set IC of integrity constraints (ICs) helps specify and maintain the correspondence between D and that reality

- Methodologies for dealing with ICs are quite general and have broad applicability
- However, in many situations databases may be inconsistent wrt. a given set of ICs

  Getting rid of violations is sometimes impossible or too complex or undesirable
- Why not accepting inconsistency, live with it, and make the best we can out of our DB?
- Database repairing and consistent query answering (CQA) are newer contributions in this direction (more coming)

  And more generally, a contribution to a newer approach to data quality problems

A database may be inconsistent!

- Poor update control, poorly designed application programs and transactions
- Delayed updates
- Legacy data on which we want to impose new constraints
- User constraints that cannot be enforced by the user
- "Informational constraints" provided by the user to the DB(MS) as extra knowledge

  The DBMS does not check or enforce them (too costly, for better performance), but just uses them

  For "semantic query optimization" (as opposed to syntactic QO), i.e. ICs are used to speed up query answering

Example: IC says there are not student numbers above 5000;

A query asks about the student with student number 5025

No need to inspect the table ... (empty answer)

- Virtual data integration systems

  Each source may be locally consistent wrt. its own ICs

  If we want ICs at the global, mediated level, there is no way to check or enforce them (no direct access to the data sources)

- ...

Characterizing Consistent Data wrt. ICs

- What are the consistent data in an inconsistent database?

  What are the consistent answers to a query posed to an inconsistent database?

- A precise definition was provided: [PODS'99]

  Intuitively, the consistent data in an inconsistent database D are invariant under all minimal ways of restoring D's consistency

  Consistent data persists across all the minimally repaired versions of the original instance: the repairs of D

Example: For the instance D that violates FD: $Name \rightarrow Salary$

Employee	*Name*	*Salary*
	page	5K
	page	8K
	smith	3K
	stowe	7K

Two possible (minimal) repairs if only deletions/insertions of whole tuples are allowed: D_1, resp. D_2

Employee	*Name*	*Salary*
	page	5K
	smith	3K
	stowe	7K

Employee	*Name*	*Salary*
	page	8K
	smith	3K
	stowe	7K

$(stowe, 7K)$ persists in all repairs: it is consistent information

$(page, 8K)$ does not (it participates in the violation of FD)

- A consistent answer to a query Q from a database D is obtained as a usual answer to Q from every possible repair of D wrt. IC
  - $Q_1 : Employee(x, y)?$

    Consistent answers: $(smith, 3K), (stowe, 7K)$
  - $Q_2 : \exists y Employee(x, y)?$

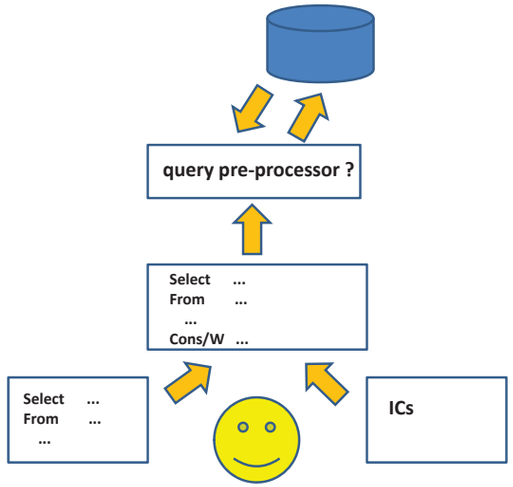
    Consistent answers: $(page), (smith), (stowe)$
- CQA may be different from traditional data cleaning!
- Repairs and CQA have become relevant in data quality

  It also provides concepts and techniques for data cleaning

  And beyond relational databases, e.g. ontology-based data access (OBDA)

- Next DBMSs should provide more flexible, powerful, and user friendlier mechanisms for dealing with semantic constraints

  In particular, they should accept and answer queries requesting for consistent data



Why not an enhanced SQL?

```
SELECT      Name, Salary
FROM        Employee
CONS/W      FD: Name -> Salary;
```

(FD not maintained by the DBMS)

- Paradigm shift: ICs are constraints on query answers, not on database states!

  A form of data cleaning wrt. IC violation at query-answering time!

- Idea: For CQA avoid or minimize computation/materialization of repairs

- For some tractable cases of CQA, query rewriting algorithms have been developed

  New query can be posed/answered as usual to/from the original DB

$$\mathcal{Q}(x, y): \ Employee(x, y) \quad \mapsto$$
$$\mathcal{Q}'(x, y): \ Employee(x, y) \wedge \neg\exists z(Employee(x, z) \wedge z \neq y)$$

```
SELECT    Name, Salary    ↦   SELECT    Name, Salary
FROM      Employee;             FROM    Employee
                                WHERE   NOT EXISTS (
                                  SELECT *
                                  FROM   Employee E
                                  WHERE  E.Name = Name AND
                                         E.Salary <> Salary);
```

(retrieves employees with their salaries for which there is no other employee with the same name, but different salary)

- This relational calculus (RC) to RC rewriting not always possible, for complexity reasons
- Evaluation of RC queries is doable in polynomial time in data complexity (i.e. in the size of the original instance)

  CQA can be $coNP$-complete (or higher) in data complexity
- Any more general and declarative approach to CQA?

- Repairs can be specified by logic programs with stable model semantics (answer-set programs)

Example: Schema $R(A, B, C, D)$, with intended FD $AB \rightarrow C$

Equivalently in RC: $\neg\exists xyz_1z_2vw(R(x, y, z_1, v) \wedge R(x, y, z_2, w) \wedge z_1 \neq z_2)$

The repair program implicitly specifies repairs, with rules with annotation constants in extra attribute, and original DB as facts

Possible deletions:

$$R'(t_1, x, y, z_1, v, \mathsf{d}) \vee R'(t_2, x, y, z_2, w, \mathsf{d}) \leftarrow R(t_1, x, y, z_1, v), R(t_2, x, y, z_2, w), z_1 \neq z_2$$

What stays in DB after deletions:

$$R'(t, x, y, z, v, \mathsf{s}) \leftarrow R(t, x, y, z, v),\ \ not\ R'(t, x, y, z, v, \mathsf{d})$$

- Stable models of the program are in 1-1 correspondence with repairs

- CQA can be done querying the program under certain answer semantics:

Example: (cont.) CQA to projection query: $\exists Y \exists W\, R(X, Y, Z, W)$?

Add query rule to the program:

$$Ans(x, z) \leftarrow R'(x, y, z, w, \mathsf{s})$$

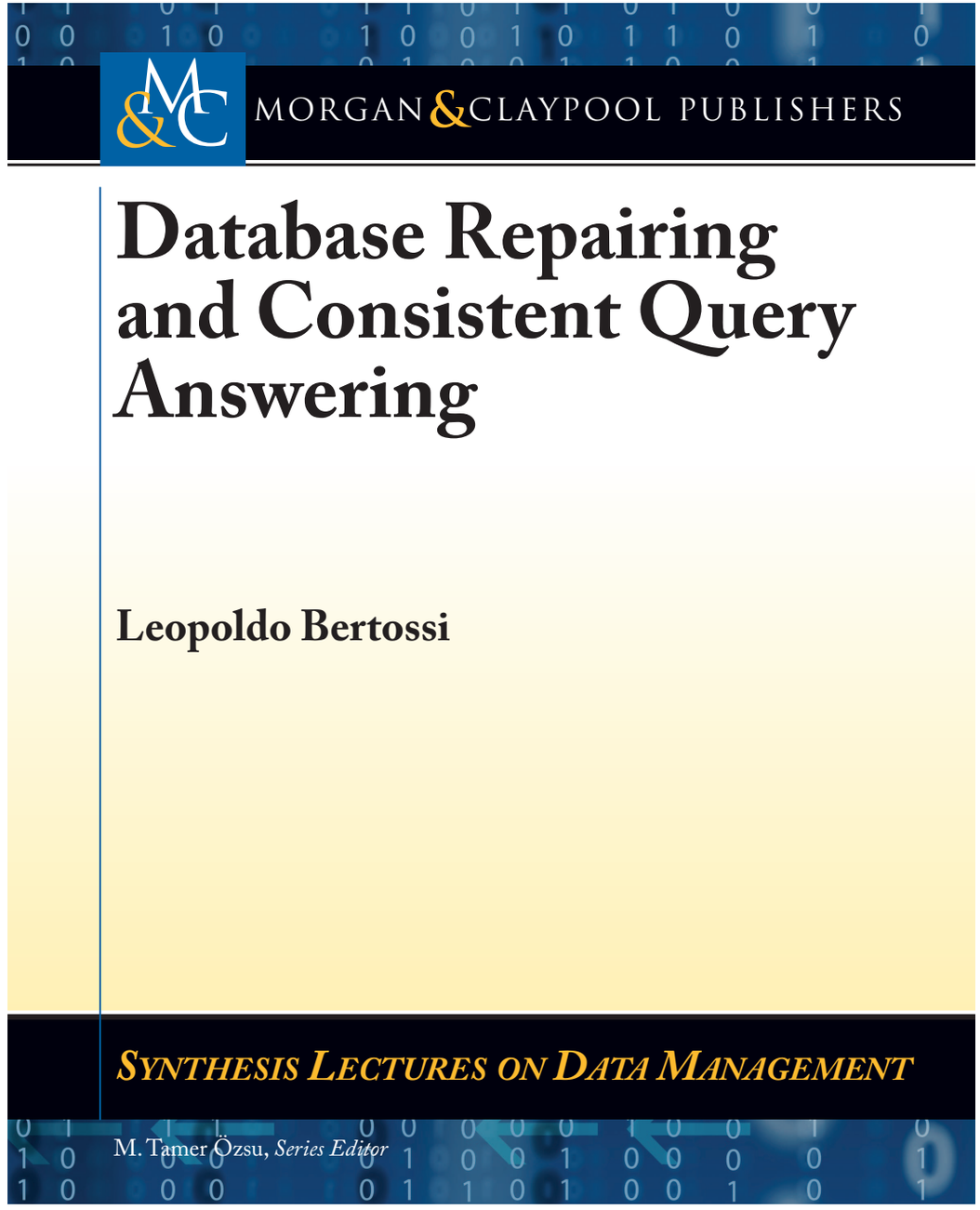
CQAs can be found in extension of auxiliary predicate Ans

- Repair programs capture higher complexity classes

  Less efficient than RC rewriting when the latter possible

- Repair programs can be optimized with magic-sets methods

- Specific, tailored, lower-complexity programs automatically obtained for lower-complexity classes of ICs


MORGAN & CLAYPOOL PUBLISHERS
Database Repairing and Consistent Query Answering
Leopoldo Bertossi
SYNTHESIS LECTURES ON DATA MANAGEMENT
M. Tamer Özsu, Series Editor

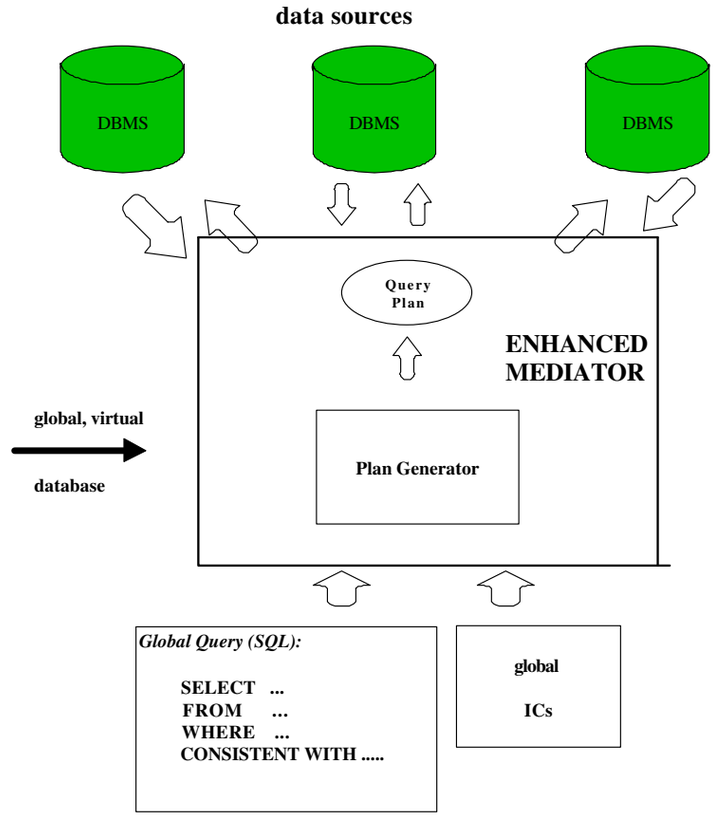
- Natural application scenario:

  Virtual data integration

  No way to enforce global ICs on the sources

  Inconsistencies have to be solved on-the-fly, at query-answering time [LNCS 3300]

- Also in peer-data exchange systems (with Nulls à la SQL) [TPLP'17]



Conclusions

- Several other repairs semantics have been investigated

  A principled approach to repair semantics selection is needed

- A complete complexity landscape for CQA under FDs and key constraints (with a caveat) [Wijsen; LNCS 8367; SIGMOD Rec.'16]

- Connections between repairs and causality and responsibility in databases [ICDT'15; TOCS 2017]

  Different repair semantics may lead to different notions of cause and responsibility

- Connections between repairs and DB updates through views [forth. IJAR]

- ICs can be conceived as declarative quality constraints on a possibly dirty (inconsistent) database (or set of query answers)

- Very much in line with classes of dependencies recently introduced and motivated by data quality/cleaning concerns

  W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2012

Example: Database relation with FDs:

FD_1: $[CC, AC, Phone] \rightarrow [Street, City, Zip]$

FD_2: $[CC, AC] \rightarrow [City]$

CC	AC	Phone	Name	Street	City	Zip
44	131	1234567	mike	mayfield	NYC	EH4 8LE
44	131	3456789	rick	crichton	NYC	EH4 8LE
01	908	3456789	joe	mtn ave	NYC	07974

FDs are satisfied, but they are "global" ICs

They may not capture natural data quality requirements ...

... those related to specific data values

- What about a *conditional functional dependency* (CFD)?

$$CFD_1: \quad [CC = 44, Zip] \rightarrow [Street]$$

The FD of $Street$ upon Zip applies when the country code is 44

Not satisfied anymore, and data cleaning may be necessary ...

- More generally, CDs are like classical ICs with a *tableau* for forced data value associations

CFD_2:

$$[CC = 44, AC = 131, Phone] \rightarrow [Street, City = \text{`}EDI', Zip]$$

When $CC = 44, AC = 131$ hold, the FD of $Street$ and Zip upon $Phone$ applies, and the city is 'EDI'

Not satisfied either ... New data quality issues detected

- CQA and database repairs have been investigated for CFDs

B. Matching Dependencies and Entity Resolution

Matching Dependencies (MDs)

- MDs are related to Entity Resolution (ER)
- ER is a classical, common and difficult problem in data cleaning

  ER is about discovering and merging records that represent the same entity in the application domain

  Detecting and getting rid of duplicates!
- Many *ad hoc* mechanisms have been proposed
- ER is fundamental for data analysis and decision making in BI
- Particularly crucial in data integration

- MDs express and generalize ER concerns

  They specify attribute values that have to be made equal under certain conditions of similarity for other attribute values

Example: Schema $R_1(X, Y), R_2(X, Y)$

$$\forall X_1 X_2 Y_1 Y_2 (R_1[X_1] \approx R_2[X_2] \longrightarrow R_1[Y_1] \doteq R_2[Y_2])$$

"When the values for attributes X_1 in R_1 and X_2 in R_2 in two tuples are similar, then the values in those two tuples for attribute Y_1 in R_1 and Y_2 in R_2 must be made equal"

(R_1 and R_2 can be same predicate)

$\approx$: Domain-dependent, attribute-level similarity relation

- MDs introduced by W. Fan et al. [PODS'08; VLDB'09]

Example:

People	Name	Phone	Addr
	peter	123456	NULL
	john	135791	sparks
	john d.	135791	sparks av.

D

MD: $People[Phone_1] \approx People[Phone_2] \rightarrow People[Addr_1] \doteq People[Addr_2]$

"every two tuples that have similar phone numbers must have their addresses identified (matched)"

In this case, the two addresses in red have to be matched, e.g.:

$\mathfrak{m}_{Addr}(sparks, sparks\ av.) := sparks\ av.$ (matching function picking most informative value)

After "applying" (enforcing) the MD, we get new instance

People	Name	Phone	Addr
	peter	123456	NULL
	john	135791	sparks av.
	john d.	135791	sparks av.

D'

- MDs could be seen as "procedural" extensions of "equality generating dependencies" (egds)

- MDs have a non-classical, dynamic semantics

  Pairs of instances (D, D') satisfy the MD

  D satisfies the antecedent, and D', the consequent, where the merging is realized — one-step satisfaction

- We may need several steps until reaching an instance where all the intended mergings are realized

  Dirty instance: $D \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots\ldots \Rightarrow D'$

  D': stable, clean instance!

  We defined the crucial "$\Rightarrow$" step (there could be several of these)

Matching Dependencies with MFs

"similar name and phone number $\Rightarrow$ identical address"

$$People[Name_1] \approx_{nm} People[Name_2] \ \wedge \ People[Phone_1] \approx_{ph} People[Phone_2]$$

$$\rightarrow \ People[Addr_1] \doteq People[Addr_2]$$

D_0	*name*	*phone*	*address*
	John Doe	(613)123 4567	Main St., Ottawa
	J. Doe	123 4567	25 Main St.

$\Downarrow$

D_1	*name*	*phone*	*address*
	John Doe	(613)123 4567	25 Main St., Ottawa
	J. Doe	123 4567	25 Main St., Ottawa

$$\mathfrak{m}_{address}(\underline{MainSt., Ottawa} \ , \ \underline{25MainSt.}) := \underline{25MainSt., Ottawa}$$

Addresses treated as strings or objects, i.e. sets of pairs attribute/value

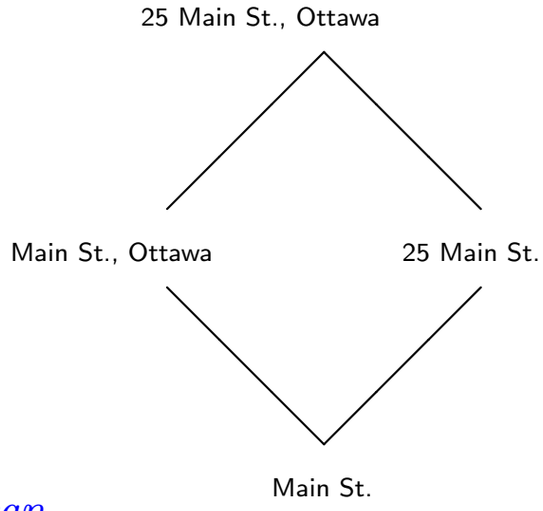
[ICDT'11, TOCS 2013]

- Matching functions induce a partial order (lattice) on attribute domains

$$a \preceq_A a' \;:\Longleftrightarrow\; \mathfrak{m}_A(a, a') = a'$$

$a \preceq_A a'$ can be thought of in terms of information contents

When MFs are applied information contents grows, and uncertainty decreases!



$D_0 \sqsubseteq D_1 \sqsubseteq \ldots \sqsubseteq D_{clean}$

- In general, there could be multiple clean instances
- The complexity on (certain) query answering can be $\mathcal{NP}$-complete in data
- Special (syntactic) cases:
  - Similarity-preserving matching functions

$$a \approx a' \Rightarrow a \approx \mathfrak{m}_A(a', a'')$$

  - Interaction-free MDs

A unique clean instance!

Computable in polynomial-time in data

Answer-Set Programs for MD-Based ER

- General MDs can be specified/enforced with *answer-set programs* (ASPs) [KR'12]

  A declarative, logic-based specification

  Non-stratified negation needed, but no disjunction

  Which matches intrinsic complexity of enforcing MDs

- Answer-sets (or stable models) in correspondence with the different final clean instances

  Program has to simulate admissible chase sequences, the most complex part

Example: An unresolved instance; similarities: $a_1 \approx a_2$, $b_2 \approx b_3$; the MF; and MDs

$R(D_0)$	A	B
t_1	a_1	b_1
t_2	a_2	b_2
t_3	a_3	b_3

$M_B(b_1, b_2, b_{12})$
$M_B(b_2, b_3, b_{23})$
$M_B(b_1, b_{23}, b_{123})$
$M_B(b_3, b_4, b_{34})$

$\varphi_1 : R[A] \approx R[A] \rightarrow R[B] \doteq R[B]$
$\varphi_2 : R[B] \approx R[B] \rightarrow R[B] \doteq R[B]$

Interacting MDs; enforcing them on D_0 results in two alternative chase sequences, and two resolved instances, D_1 and D'_2

D_0	A	B
t_1	a_1	b_1
t_2	a_2	b_2
t_3	a_3	b_3

$\Rightarrow_{\varphi_1}$

D_1	A	B
t_1	a_1	b_{12}
t_2	a_2	b_{12}
t_3	a_3	b_3

D_0	A	B
t_1	a_1	b_1
t_2	a_2	b_2
t_3	a_3	b_3

$\Rightarrow_{\varphi_2}$

D'_1	A	B
t_1	a_1	b_1
t_2	a_2	b_{23}
t_3	a_3	b_{23}

$\Rightarrow_{\varphi_1}$

D'_2	A	B
t_1	a_1	b_{123}
t_2	a_2	b_{123}
t_3	a_3	b_{23}

The chase sequences and clean instances can be specified with a generic ASP:

1. $R'(t_1, a_1, b_1).\ R'(t_2, a_2, b_2).\ R'(t_3, a_3, b_3).$ (plus M_B facts)

2. $Match_{\varphi_1}(T_1, X_1, Y_1, T_2, X_2, Y_2) \ \vee$
$NotMatch_{\varphi_1}(T_1, X_1, Y_1, T_2, X_2, Y_2) \ \leftarrow$
$R'(T_1, X_1, Y_1),\ R'(T_2, X_2, Y_2),\ X_1 \approx X_2,\ Y_1 \neq Y_2.$
$Match_{\varphi_2}(T_1, X_1, Y_1, T_2, X_2, Y_2) \ \vee$
$NotMatch_{\varphi_2}(T_1, X_1, Y_1, T_2, X_2, Y_2) \ \leftarrow$
$R'(T_1, X_1, Y_1),\ R'(T_2, X_2, Y_2),\ Y_1 \approx Y_2,\ Y_1 \neq Y_2.$

$Match_{\varphi_i}(T_1, X_1, Y_1, T_2, X_2, Y_2) \ \leftarrow$
$Match_{\varphi_i}(T_2, X_2, Y_2, T_1, X_1, Y_1). \quad (i \in \{1, 2\})$

$OldVersion_R(T_1, \bar{Z}_1) \ \leftarrow\ R'(T_1, \bar{Z}_1),\ R'(T_1, \bar{Z}'_1),$
$\bar{Z}_1 \preceq \bar{Z}'_1,\ \bar{Z}_1 \neq \bar{Z}'_1.$

$\leftarrow\ NotMatch_{\varphi_i}(T_1, X_1, Y_1, T_2, X_2, Y_2),$
$not\ OldVersion_R(T_1, X_1, Y_1),$
$not\ OldVersion_R(T_2, X_2, Y_2). \quad (i \in \{1, 2\})$

3. $R'(T_1, X_1, Y_3) \ \leftarrow\ Match_{\varphi_1}(T_1, X_1, Y_1, T_2, X_2, Y_2),$
$M_B(Y_1, Y_2, Y_3).$
$R'(T_1, X_1, Y_3) \ \leftarrow\ Match_{\varphi_2}(T_1, X_1, Y_1, T_2, X_2, Y_2),$
$M_B(Y_1, Y_2, Y_3).$

4. $Prec(T_1, X_1, Y_1, T_2, X_2, Y_2, T_1, X_1, Y'_1, T_3, X_3, Y_3) \ \leftarrow$
$Match_{\varphi_i}(T_1, X_1, Y_1, T_2, X_2, Y_2),$
$Match_{\varphi_j}(T_1, X_1, Y'_1, T_3, X_3, Y_3),$
$Y_1 \preceq Y'_1,\ Y_1 \neq Y'_1. \quad (i, j \in \{1, 2\})$

5. $Prec(T_1, X_1, Y_1, T_2, X_2, Y_2, T_1, X_1, Y_1, T_3, X_3, Y_3) \ \leftarrow$
$Match_{\varphi_i}(T_1, X_1, Y_1, T_2, X_2, Y_2),$
$Match_{\varphi_j}(T_1, X_1, Y_1, T_3, X_3, Y_3),\ M_B(Y_1, Y_3, Y_4),$
$Y_1 \neq Y_4. \quad (i, j \in \{1, 2\})$

6. $Prec(T_1, \bar{Z}_1, T_2, \bar{Z}_2, T_1, \bar{Z}_1, T_2, \bar{Z}_2) \ \leftarrow$
$Match_{\varphi_i}(T_1, \bar{Z}_1, T_2, \bar{Z}_2). \quad (i \in \{1, 2\})$

$\leftarrow Prec(T_1, \bar{Z}_1, T_2, \bar{Z}_2, T_1, \bar{Z}'_1, T_3, \bar{Z}_3),$
$Prec(T_1, \bar{Z}'_1, T_3, \bar{Z}_3, T_1, \bar{Z}_1, T_2, \bar{Z}_2),$
$(T_1, \bar{Z}_1, T_2, \bar{Z}_2) \neq (T_1, \bar{Z}'_1, T_3, \bar{Z}_3).$

$\leftarrow Prec(T_1, \bar{Z}_1, T_2, \bar{Z}_2, T_1, \bar{Z}'_1, T_3, \bar{Z}_3),$
$Prec(T_1, \bar{Z}'_1, T_3, \bar{Z}_3, T_1, \bar{Z}''_1, T_4, \bar{Z}_4),$
$not\ Prec(T_1, \bar{Z}_1, T_2, \bar{Z}_2, T_1, \bar{Z}''_1, T_4, \bar{Z}_4).$

7. $R^c(T_1, X_1, Y_1) \ \leftarrow R'(T_1, X_1, Y_1),$
$not\ OldVersion_R(T_1, X_1, Y_1).$

Non-stratified negation in 6., which capture admissible chase sequences

- ASP has two stable models, from them we can read off the two clean instances D_1, D_2'

  With the following sets of atoms for the corresponding resolved instances:

$$M_1 = \{..., R^c(t_1, a_1, b_{12}), R^c(t_2, a_2, b_{12}), R^c(t_3, a_3, b_3)\}$$

$$M_2 = \{..., R^c(t_1, a_1, b_{123}), R^c(t_2, a_2, b_{123}), R^c(t_3, a_3, b_{23})\}$$

- Resolved (clean) query answers can be obtained by querying the ASP under the certain (skeptical) semantics (returns what is true in all models)

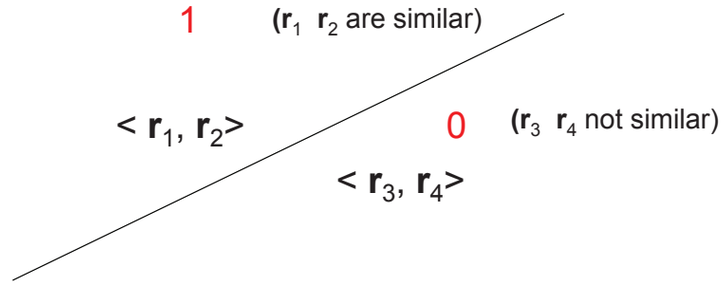
  Maybe not necessary to compute and materialize all clean instances

An Application: Blocking for Duplicate Detection

- A classification problem: $\mathbf{r_1} = \langle a_1, \ldots, a_n \rangle$ vs. $\mathbf{r_2} = \langle a'_1, \ldots, a'_n \rangle$

  In principle, every two records have to be compared, and classified

  This can be costly ...



- Reduce large number of two-record comparisons using blocking techniques

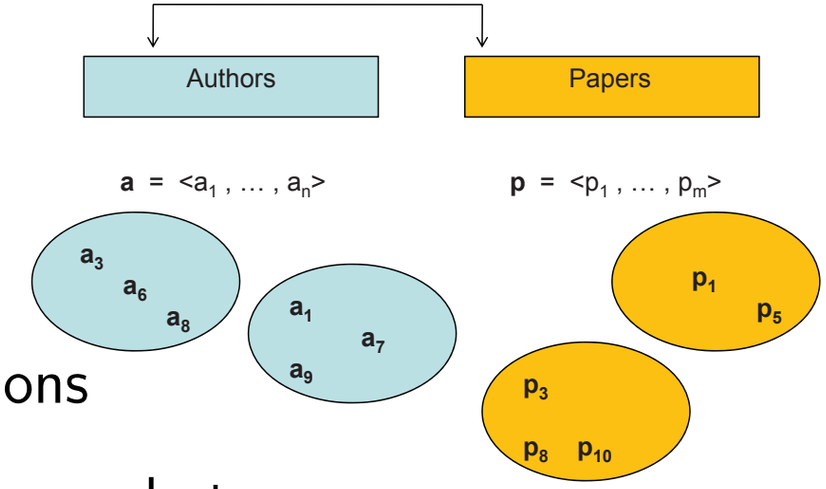
  Only records within the same block values are compared

  Any two records in different blocks will never be duplicates

- Better: Apply blocking with additional semantics or domain knowledge

Example: "author" and "paper" records

Group author records based on similarities of authors' names and affiliations



Author records $\mathbf{a_1}, \mathbf{a_2}$ may have similar names, but not similar affiliations

$\mathbf{a_1}, \mathbf{a_2}$ are authors of papers $\mathbf{p_1}, \mathbf{p_2}$, resp., but $\mathbf{p_1}, \mathbf{p_2}$ have been put in the same block of papers

Semantic knowledge: *"If two papers are in the same block, their authors with similar names should be in the same block"*

So, assign $\mathbf{a_1}, \mathbf{a_2}$ to the same block

Collective blocking! of author and paper entities, separately, but based on relational closeness (not only local, attribute-level similarities)

Relational MDs

How can we capture this kind of additional semantic knowledge?

- An extended, relational kind of MD: [IJAR'17]

$$Author(x_1, y_1, \underline{bl_1}) \ \wedge \ Paper(y_1, z_1, bl_3) \ \wedge \ Author(x_2, y_2, \underline{bl_2}) \ \wedge$$
$$Paper(y_2, z_2, bl_3) \ \wedge \ x_1 \approx_1 x_2 \ \wedge \ z_1 \approx_2 z_2 \ \longrightarrow \ \underline{bl_1} \doteq \underline{bl_2}$$

- MDs originally for merging, but now these are blocking MDs!
- Similarities (LHS) and mergings (RHS) use domain-dependent relations and matching functions (MFs), resp.

Example: Author and Paper entities ("*R. Smith*" ≈ "*MR. Smyth*")

Author	*Name*	*Affiliation*	*PaperID*	*Bl#*
12	*R. Smith*	*MBA, UCLA*	1	12
13	*MR. Smyth*	*MBA*	2	13
14	*J. Doe*	*MBA, UCLA*	3	14

Paper	*Title*	*Year*	*AuthorID*	*Bl#*
1	*Illness in Africa*	1990	12	2
2	*Illness in West Africa*	90	13	2

- Records have unique, global ids (positive integers)
- Initial block number $Bl\#$ for a record is its id
- Two records are forced to go into same block by enforcing equality of their block numbers

  Use MDs with a MF: $\mathfrak{m}_{Bl\#}(b_i, b_j) := b_i \quad \text{if} \quad b_j \leq b_i$

"Group two author entities into same block if they have similar names and affiliations or they have similar names and their corresponding papers are in same block"

$$m_1: \quad Author(a_1, x_1, y_1, p_1, \underline{b_1}) \wedge Author(a_2, x_2, y_2, p_2, \underline{b_2}) \wedge x_1 \approx x_2 \ \wedge \ y_1 \approx y_2 \ \rightarrow \ \underline{b_1} \doteq \underline{b_2}$$

$$m_2: \quad Author(a_1, x_1, y_1, p_1, \underline{b_1}) \ \wedge Author(a_2, x_2, y_2, p_2, \underline{b_2}) \ \wedge x_1 \approx x_2 \wedge Paper(p_1, x_1', y_1', a_1, b_3) \ \wedge \ Paper(p_2, x_2', y_2', a_2, b_3) \ \rightarrow \ \underline{b_1} \doteq \underline{b_2}$$

Enforcing results in DB with two author-blocks: $\{12, 13\}, \ \{14\}$

Enforcing Blocking MDs and Beyond

- In our application we wanted -among other tasks- to enforce blocking MDs on relational DBs with (stratified) Datalog

  Implemented as *LogiQL* by LogicBlox `http://www.logicblox.com/`

  Stratified Datalog programs can be computed in PTIME in data

- General ASP, as provably needed for general MDs, not supported by *LogiQL*

  PTIME vs. NP-complete

- Could we do with less than ASP for blocking MDs?

In more general terms:

- We confronted the problem of identifying combinations of syntactic classes of relational MDs with instances, and similarities for which:
  - Recognition can be done efficiently
  - A unique resolved instance exists
  - Computable in PTIME in the size of the instance
  - Computable with a Datalog program
  - Specialized "uniformly and syntactically" from the general ASP
- We identified some classes
- Blocking sets of relational MDs have those properties

  A single "blocking solution" can be obtained applying stratified Datalog

Example: (continued) This time with similarities: $a_1 \approx a_2$, $b_3 \approx b_4$

- The combination falls under the "Single Resolved Instance" combination class
- The ASP can be automatically and syntactically rewritten

Now Datalog:

$$
\begin{aligned}
&1.\ R(t_1,a_1,b_1).\ R(t_2,a_2,b_2).\ R(t_3,a_3,b_3).\ R(t_4,a_4,b_4).\\
&2.\ Match_{\varphi_1}(T_1,X_1,Y_1,T_2,X_2,Y_2) \leftarrow R(T_1,X_1,Y_1),\\
&\qquad\qquad R(T_2,X_2,Y_2),\ X_1 \approx X_2,\ Y_1 \neq Y_2.\\
&\quad Match_{\varphi_2}(T_1,X_1,Y_1,T_2,X_2,Y_2) \leftarrow R(T_1,X_1,Y_1),\\
&\qquad\qquad R(T_2,X_2,Y_2),\ Y_1 \approx Y_2,\ Y_1 \neq Y_2.\\
&\quad OldVersion(T_1,X_1,Y_1) \leftarrow R(T_1,X_1,Y_1),\\
&\qquad\qquad R(T_1,X_1,Y_1'),\ Y_1 \preceq Y_1',\ Y_1 \neq Y_1'.\\
&3.\ R(T_1,X_1,Y_3) \leftarrow Match_{\varphi_1}(T_1,X_1,Y_1,T_2,X_2,Y_2),\\
&\qquad\qquad M_B(Y_1,Y_2,Y_3).\\
&\quad R(T_1,X_1,Y_3) \leftarrow Match_{\varphi_2}(T_1,X_1,Y_1,T_2,X_2,Y_2),\\
&\qquad\qquad M_B(Y_1,Y_2,Y_3).\\
&7.\ R^c(T_1,X_1,Y_1) \leftarrow R(T_1,X_1,Y_1),\\
&\qquad\qquad not\ OldVersion(T_1,X_1,Y_1).
\end{aligned}
$$

Only stratified negation, in 7.

Conclusions

- Datalog, a rule-based extension of relational algebra/calculus/databases, enables declarative and executable specifications in DBs

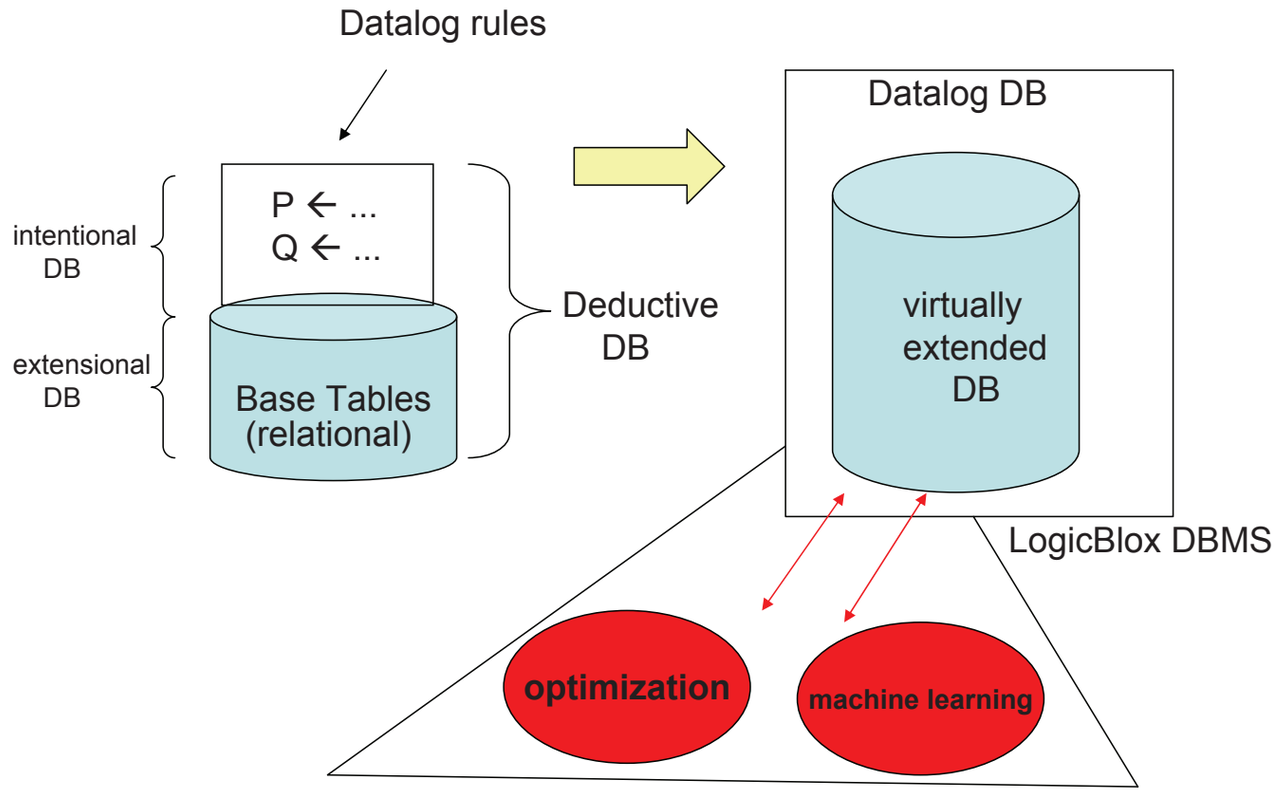
  Around since the early 80s, for long time used mostly in DB research, it has experienced a revival in the last few years!

- Our *ERBlox* system for ER was developed in collaboration with LogicBlox

  It combines MDs (for blocking and merging), *LogiQL*, and SVMs (for the classifier)

  All on top of *LogiQL*, the Datalog version of the *LogicBlox* platform

- High-level goal is extend LogiQL with ML and optimization methods


Datalog rules
P ← ...
Q ← ...
intentional DB
extensional DB
Base Tables (relational)
Deductive DB
Datalog DB
virtually extended DB
LogicBlox DBMS
optimization
machine learning

C.1. Contexts in Data Quality

Our Initial Motivation: Contexts

Example: Doctor requires temperatures taken with oral thermometer, expecting data to correspond to this requirement

TempNoon@Ward

Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

Here quality refers to the origin and use of data rather that about satisfaction of ICs

Table has no elements for this assessment

An external context can provide them

- Given a data source, we may want to:
  - Analyze, understand, make sense of the data, etc.
  - Assess the data quality
- All this is a formal setting in which the data is embedded
- Contexts were introduced for data quality assessment and quality-data extraction [VLDB'10 BIRTE WS]

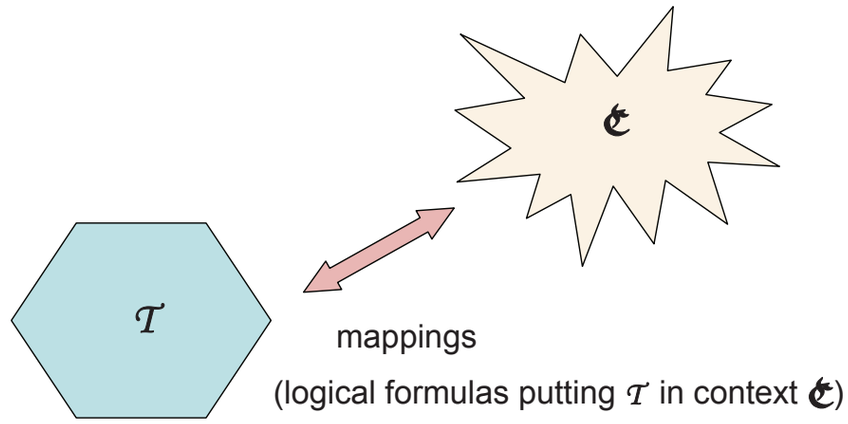
  Specified as a separate relational database or a (virtual) data integration system
- D can be mapped into the context
- Quality criteria imposed at contextual level

  Depending on the mapping and context's ingredients

Contexts: A Vision

- A logical theory $\mathcal{T}$ is the one that has to be "put in context"

  A relational database D can be seen as a logical theory $\mathcal{T}$, e.g. Reiter'slogical reconstruction of a relational DB

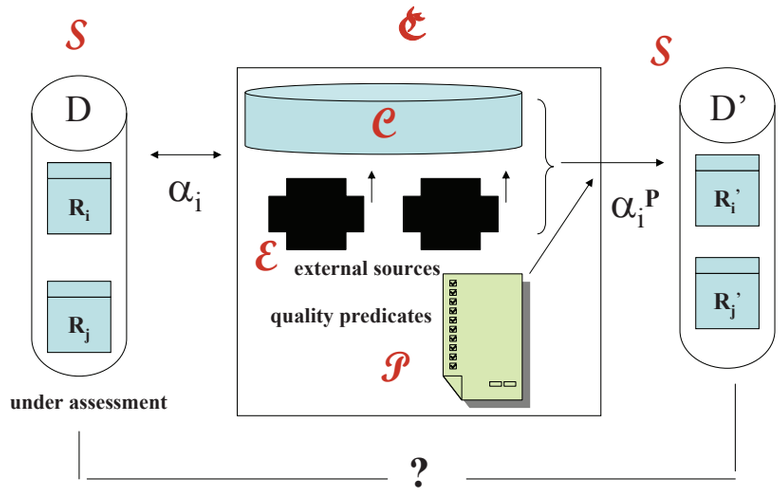


- The context is another logical theory, $\mathcal{C}$

  An ontology, a virtual data integration system (also expressible in FOL), etc.

- Connection between $\mathcal{T}$ and $\mathcal{C}$ is established through "bridge" predicates and mappings

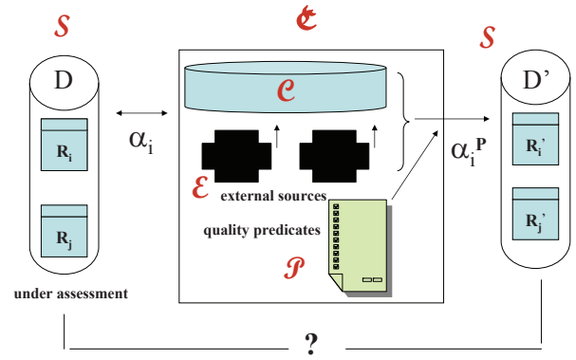
A Simple Model of Context



- Instance D is under assessment
- On RHS, also schema $\mathcal{S}$ (or copy $\mathcal{S}'$)
- Context $\mathfrak{C}$ as virtual (semi)materialized integration system

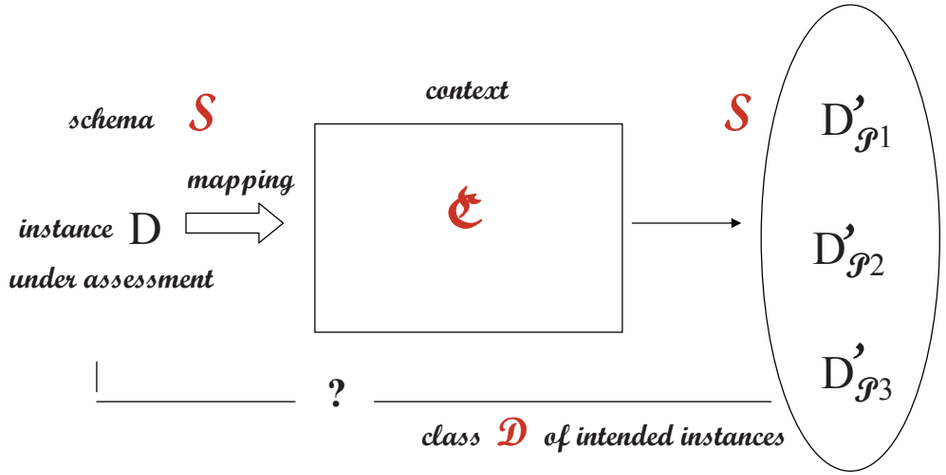
- The α_i are the mappings, like in VDISs or data exchange
- The C_i are contextual predicates/relations
- Mappings to quality predicates/relations P_i and possibly external sources E_i
- D' contains "ideal" contents for relations in D, as views

- Predicates in D' can be materialized through data in the R_i and additional massage via $\mathcal{C}$
  (mapping composition at work)



- Quality assessment of D can be done by comparing its contents with D' (there are some measures)

- More generally, through the context, alternative, admissible, clean versions of D can be specified, computed, compared (with each other and D), queried, etc.



A class $\mathcal{D}$ of admissible contextual instances D' for schema $\mathcal{S}$

- Quality-aware (QA) query answering about (or from) $\mathcal{S}$ can be done on top of D'
- Techniques for query answering in virtual integration can be applied, specially if D' or $(\mathcal{D})$ are not materialized

  Quality assessment becomes particular case of QA

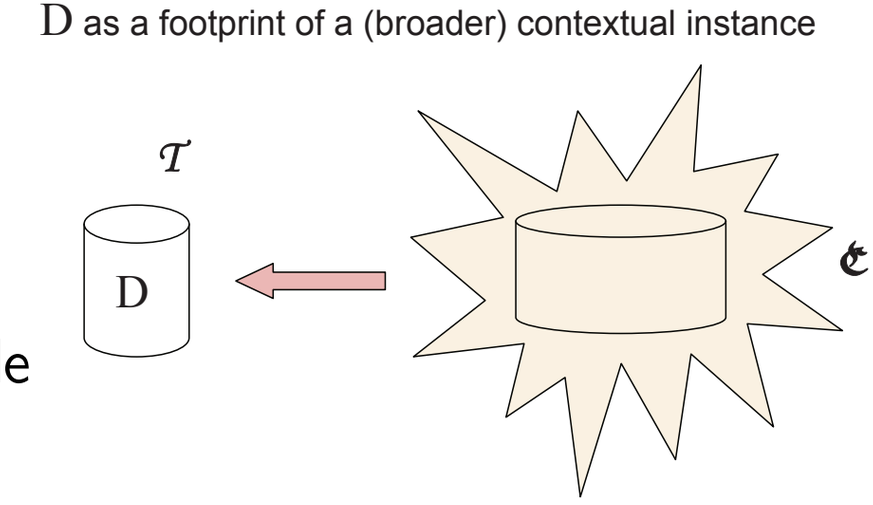
Different cases emerge ... Among them:

http://arxiv.org/abs/1608.04142

(A)

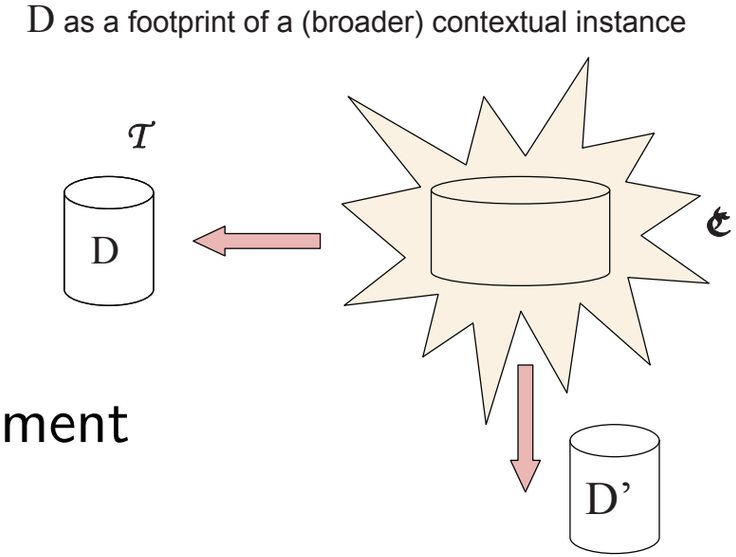
Data in $\mathfrak{C}$ (including D) is analyzed or cleaned

According to additional data D^c available in or accessible from $\mathfrak{C}$; and quality criteria defined in $\mathfrak{C}$



A new version D' of D is obtained

Can be compared with D for quality assessment



Example: (the simple case) A contextual instance $Measurements$

Initial table $TempNoon$ (the R in D on page 47) is a view (footprint) of $Measurements$ with mapping α

Measurements (contextual)

	Patient	Value	Time	Date	Instr
1	T. Waits	37.8	11:00	Sep/5	Oral Therm.
2	T. Waits	38.5	11:45	Sep/5	Tympanal Therm.
3	T. Waits	38.2	12:10	Sep/5	Oral Therm.
	...	...	...	...	...
4	T. Waits	110/70	11:00	Sep/6	BPM
5	T. Waits	38.1	11:50	Sep/6	Oral Therm.
6	T. Waits	38.0	12:15	Sep/6	Oral Therm.
	...	...	...	...	...
7	T. Waits	37.6	10:50	Sep/7	Tympanal Therm.
8	T. Waits	120/70	11:30	Sep/7	BPM
9	T. Waits	37.9	12:15	Sep/7	Oral Therm.

$$TempNoon(p, v, t, d) \longleftarrow Measurements(p, v, t, d, i)$$

Here, $\mathcal{D} = \{D'\}$, a single admissible contextual instance

Impose quality requirements: (the R' and α^P above)

$$TempNoon'(p, v, t, d) \longleftarrow Measurements(p, v, t, d, i),$$
$$11{:}30 \ \leq \ t \ \leq \ 12{:}30, \ i = oral\ therm$$

Here, $R'(D') \subseteq R(D)$, and set-difference $R(D) \smallsetminus R'(D')$ indicates how initial $R(D)$ departs from quality instance $R'(D')$

$$TempNoon'(D') \subsetneqq TempNoon(D)$$

(in other cases, maybe $\Delta(R(D), R'(D'))$)

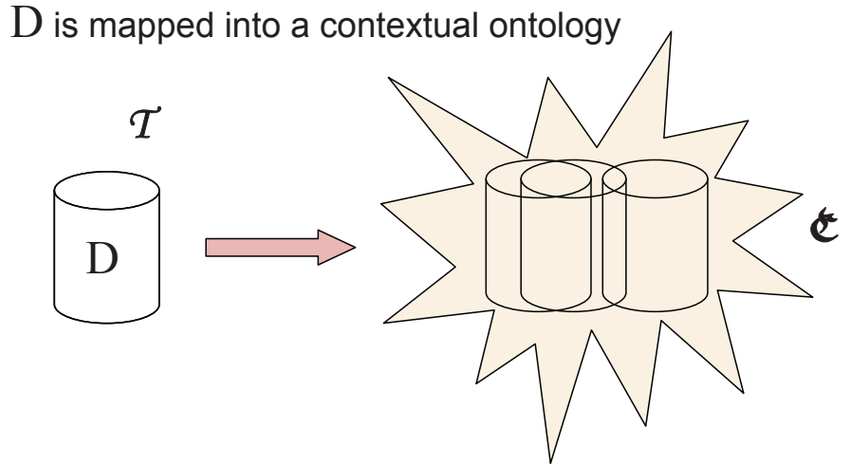
Quality query answering? (conjunctive queries)

$$\begin{array}{ccc} \mathcal{Q} \in L(\mathcal{S}) & \mapsto\mapsto\mapsto & \mathcal{Q}' \in L(\mathcal{S}') \\ \swarrow & (R \mapsto R') & \searrow \\ R(D) & & R'(D') \\ & & \underbrace{\qquad\qquad} \\ & & \text{Or} \quad \downarrow \end{array}$$

Instead of RHS in red, view unfolding: $\mathcal{Q}' \mapsto \mathcal{Q}'' \in L(\mathfrak{C}) \to D^c$

As expected: $\mathcal{Q}''(D^c) \subseteq \mathcal{Q}(D)$ (monotone query and additional conditions)

(B)



In principle several versions of D can be obtained at the contextual level

Depending on the mapping, assumptions about contextual data as source (completeness?), availability of (partial) data at the context, etc.

Example: Now we have initial instance D, but an incomplete or missing contextual instance

We map D to the contextual schema, imposing there the quality requirements (in a language associated to $\mathfrak{C}$)

Again: $TempNoon(p, v, t, d) \longleftarrow Measurements(p, v, t, d, i)$

Data are in $TempNoon(D)$, no (or some) data for $Measurements$

Instrument i might be obtained from additional contextual data

As in LAV: Possible several admissible instances for (global, mediated) $Measurements$, then the same for D' in $\mathcal{D}$

With quality requirements:

$$TempNoon'(p, v, t, d) \longleftarrow Measurements(p, v, t, d, i), 11{:}30 \leq t \leq 12{:}30,\ i = oral\ therm$$

- Possible several instances for schema S', the $D' \in \mathcal{D}$, with $D' \subseteq D$
- Quality of D? Measured as distance to the class $\mathcal{D}$ of quality versions

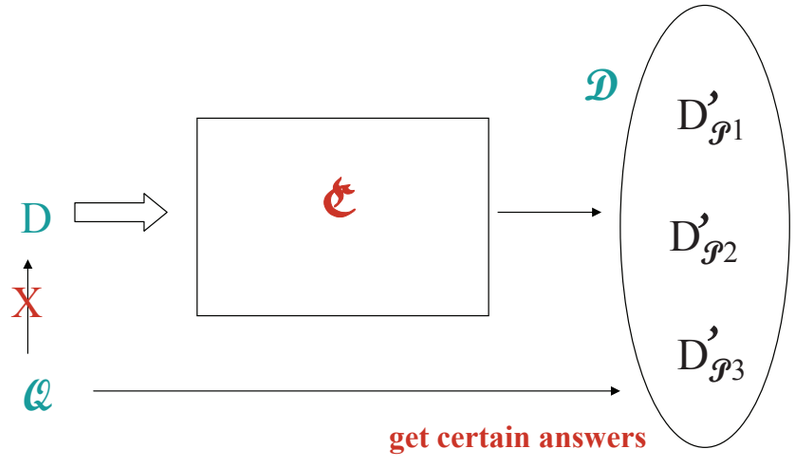
  A possible quality measure: $q(D) := \frac{|D| - max\{|D'| : D' \in \mathcal{D}\}}{|D|}$

  Computation, approximation?
- Quality criteria imposed at the contextual level
- This framework opens the ground for "quality query answering"

  Certain answers on $\mathcal{D}$ (e.g. query rewriting via rule inversion)
- Quality data extraction via quality query answering

  In particular, clean version of table R? Pose query $\mathcal{Q}(\bar{x}) : R'(\bar{x})$?

- Given a query $\mathcal{Q}$ posed to original, dirty D
- Quality answers from D to $\mathcal{Q}$
  are certain wrt. class $\mathcal{D}$



- Issues:
  - Data quality assessment via quality query answering
  - Data cleaning vs. quality query answering
  - Computation of quality answers
  - Extends repairs and CQA: ICs go into the context
    Different kinds of quality constraints could go there

C.2. Ontological Multidimensional Contexts and Data Quality

Multidimensional Contexts

Example: (revisited) Doctor requires temperatures taken with oral thermometer, expecting data to correspond to this requirement

TempNoon@Ward

Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

- Data dimensions were not introduced above

  They are crucial in many data analysis and management problems

  We may be missing "dimensions" above, something intrinsically "contextual"

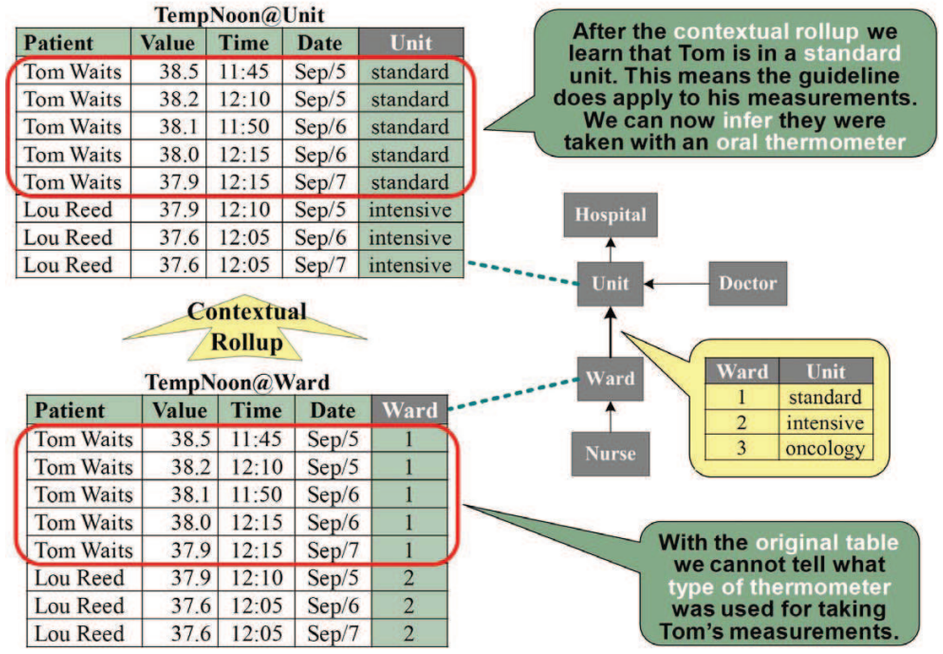
- The context could be a (multi-)dimensional database, or a dimensional ontology

  The Ontological Multidimensional Data Model (OMD model) provides formal contexts for the above tasks, with explicit dimensions

A MD data model/instance

A hospital guideline

As a rule or a constraint



"Take patients' temperatures in standard care units with oral thermometers"

Can be taken advantage of through/after upward navigation in the hierarchical, dimensional hospital structure

- A MD context would enable contextual, dimensional navigation, roll-up & drill-down

  To access and generate missing data at certain levels (as in example above)

- Idea: Embed Hurtado-Mendelzon (HM) MD data model in contexts

- Go beyond: Enrich it with additional, dimension-related data, rules and constraints

  An ontological, multidimensional context!

Ontological Contexts with Dimensions

New ingredients in MD contexts: (RuleML'15, JDQ)

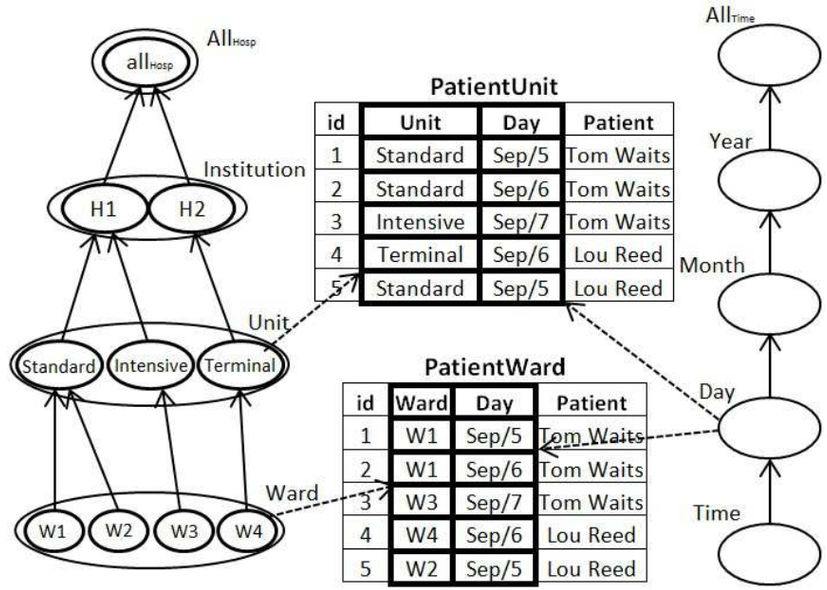
- A (relational reconstruction of) the HM model
- Categorical relations: Generalize fact tables

  Not necessarily numerical values, linked to different levels of dimensions, possibly incomplete
- Dimensional rules: generate data where missing, enable navigation
- Dimensional constraints: on (combinations of) categorical relations, involving values from dimension categories

Example:

- Categories *Ward* and *Unit* in *Hospital* dimension
- $UnitWard(unit, ward)$: parent/child relation
- *PatientWard*: categorical relation

  Ward and *Day* categorical attributes take values from categories



- Categorical relations are subject to dimensional constraints
- Need rules for dimensional navigation

What language to express all this? Datalog$^{\pm}$ (Gottlob et al., ∞)

$\mathbf{Datalog^{\pm}}$ MD Ontologies

Dimensional Constraints:

- A referential constraint restricting units in $PatientUnit$ to elements in the $Unit$ category, as a negative constraint

$$\bot \leftarrow PatientUnit(\underline{u, d}; p), \neg Unit(u)$$

- *"All thermometers used in a unit are of the same type"* :

$$t = t' \leftarrow Thermometer(\underline{w, t}; n), Thermometer(\underline{w', t'}; n'),$$
$$UnitWard(u, w), UnitWard(u, w')$$

An EGD

$Thermometer(ward, thermometertype; nurse)$ is categorical relation, t, t' for categorical attributes

- *"No patient in intensive care unit on August /2005"*:

$$\bot \leftarrow PatientWard(\underline{w, d}; p), UnitWard(\texttt{Intensive}, w),$$
$$MonthDay(\texttt{August/2005}, d)$$

An NC

Dimensional Rules:

- Data in $PatientWard$ generate data about patients for higher-level categorical relation $PatientUnit$

$$PatientUnit(\underline{u, d}; p) \leftarrow PatientWard(\underline{w, d}; p),\ UnitWard(u, w)$$

To navigate from $PatientWard.Ward$ up to $PatientUnit.Unit$ via $UnitWard$

A TGD

Once at the level of *Unit*, take advantage of guideline (a rule):

"Temperatures of patients in a standard care unit are taken with oral thermometers"

Data at $Unit$ level that can be used there and at $Ward$ level

- Data in categorical relation *WorkingSchedules* generate data in categorical relation *Shifts*

WorkingSchedules

Unit	Day	Nurse	Type
Intensive	Sep/5	Cathy	cert.
Standard	Sep/5	Helen	cert.
Standard	Sep/6	Helen	cert.
Standard	Sep/9	Mark	non-c.

Shifts

Ward	Day	Nurse	Shift
W4	Sep/5	Cathy	night
W1	Sep/6	Helen	morning
W4	Sep/5	Susan	evening

$$\exists z\ Shifts(w, d; n, z) \leftarrow WorkingSchedules(u, d; n, t), UnitWard(u, w)$$

Captures guideline: *"If a nurse works in a unit on a specific day, she has shifts in every ward of that unit on the same day"*

Existential variable z for missing values for the non-categorical *shift* attribute

Rule for downward- navigation and value invention, with join via categorical attribute between categorical and parent-child predicate

Properties of MD Ontologies

- With reasonable and natural conditions, Datalog$^{\pm}$ MD ontologies become weakly-sticky Datalog$^{\pm}$ programs [Cali et al., AIJ'12]

  Important that join variables in TGDs are for categorical attributes (with values among finitely many category members)

- The chase (that enforces TGDs) may not terminate

  Weak-Stickiness guarantees tractability of conjunctive QA: only a "small", initial portion of the chase has to be queried

  Boolean conjunctive QA is tractable for weakly-sticky (WS) Datalog$^{\pm}$ ontologies

- Separability condition on the (good) interaction between TGDs and EGDs becomes application dependent

  If EGDs have categorical head variables (as in page 66), separability holds

  Separability guarantees decidability of conjunctive QA, etc.

Next goals were: [RR'16]

(a) Develop a practical QA algorithm for WS Datalog$^{\pm}$

(b) Optimization of ontologies (as programs)

Query Answering on WS MD-Ontologies

- There was a non-deterministic PTIME algorithm for WS Datalog$^{\pm}$ (Cali et al., AIJ'12)
- Our goal was to develop a practical chase-based QA algorithm
- Apply magic-sets techniques to optimize QA

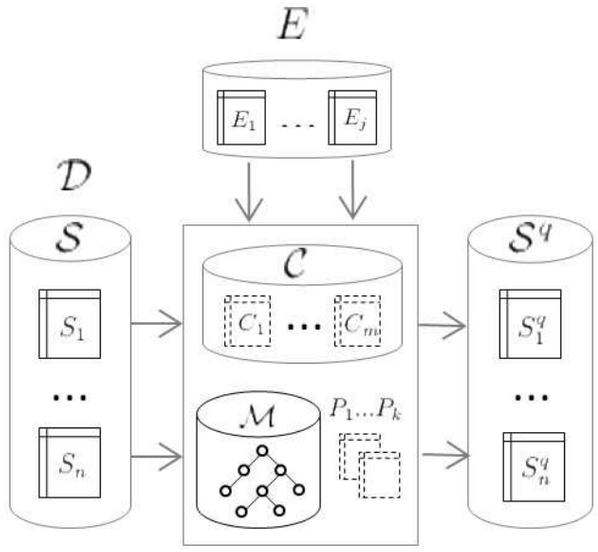
  There is such a technique (MS) available for (a class of) "existential programs" ($\exists$-Datalog) (Alviano et al., Datalog 2.0'12)
- WS Datalog$^{\pm}$ is not closed under MS
- We extended WS Datalog$^{\pm}$ to a still tractable class of *Joint-Weakly-Sticky* programs, which is closed under magic sets

  We proposed a QA algorithm

MD Contexts and Quality QA

The Datalog$^{\pm}$ MD ontology $\mathcal{M}$ becomes part of the context for data quality assessment

[RuleML'15, forth. ACM JDQ]



Original instance $\mathcal{D}$ to be assessed/cleaned through the context

By mapping $\mathcal{D}$ into the contextual schema/instance $\mathcal{C}$

- Contextual predicates C_i
- Predicates P_i specifying single quality requirements
- $\mathcal{S}^q$ copy of schema $\mathcal{S}$: S_i^q clean version of original S_i, specified using $\mathcal{C}, \mathcal{P}$ and $\mathcal{M}$

We want quality answers to the query about Tom's temperatures:

$$\mathcal{Q}(t, p, v) \leftarrow Measurements(t, p, v), p = \texttt{Tom Waits},$$
$$\texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.$$

This query does not capture quality requirements:

"Body temperatures of *Tom Waits* for *September 5* around noon taken by a *certified nurse* with a thermometer of *brand B1*"

Table *Measurements* does not contain information about nurses or thermometers

Contextual data must be taken into account, such as categorical relation *PatientUnit* and the guideline

"Temperature measurement for patients in a standard care unit are taken with thermometers of brand B1"

According to the general contextual approach DQA, table (or better predicate) $Measurement$ has to be logically connected to the context

As a "footprint" of a "broader" contextual, possibly virtual relation, given or defined in the context

One with information about thermometer brands (b) and nurses' certification status (y):

$$Measurement'(t, p, v, y, b) \leftarrow Measurement^c(t, p, v), \\ TakenByNurse(t, p, n, y), \\ TakenWithTherm(t, p, b)$$

$Measurement^c$ is contextual version of $Measurement$ (e.g. the latter mapped into the context)

Quality measurements data obtained as quality answers as in usual OBDA: (no need for computation of clean instances)

$$Measurement^q(t,p,v) \leftarrow Measurement'(t,p,v,y,b), y = \texttt{Certified},\ b = \texttt{B1}$$

Auxiliary, quality predicates above defined in terms of ontology:

$$\begin{aligned} TakenByNurse(t,p,n,y) &\leftarrow WorkingSchedules(u,d;n,y), \\ &\quad DayTime(d,t), PatientUnit(u,d;p) \\ TakenWithTherm(t,p,b) &\leftarrow PatientUnit(u,d;p), \\ &\quad DayTime(d,t), b = \texttt{B1}, u = \texttt{Standard} \end{aligned}$$

($DayTime$ is parent/child relation in *Time* dimension)

The second definition captures the guideline above

To obtain quality answers to the original query, pose a new query:

$$\mathcal{Q}^q(t,p,v) \leftarrow Measurements(t,p,v)^q,\ p = \texttt{Tom Waits},$$
$$\texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.$$

Answering it triggers dimensional navigation, when requesting data for categorical relations $PatientUnit$ and $WorkingSchedules$

Final Discussion

- Datalog$^{\pm}$ is an expressive and computationally nice family of existential programs (with constraints)
- We have used Datalog$^{\pm}$ to create multidimensional ontologies

  They can be seen as logic-based, relational extensions of MD DBs
- The OMD data model considerably extends the HM MD data model

  OMD includes general TGDs, EGDs and NCs

  A (relational reconstruction of the) HM model, data and queries are seamlessly integrated into a uniform logico-relational framework

- The usual constraints considered in the HM model are specific for the dimensional structure of data

  Most prominently, to guarantee summarizability (i.e. correct aggregation, no double-counting)

  In the HM model we find constraints enforcing strictness and homogeneity

  Strictness: Every category elements rolls-up to a single element in a parent category

  In OMD can be expressed by EGDs

  Homogeneity: Category elements have parent elements in parent categories

  In OMD can be expressed by TGDs

- OMD supports general, possibly incomplete categorical relations

  Not only complete fact tables linked to bottom categories

- Our MD ontologies belong to well-behaved classes of Datalog$^{\pm}$
- We proposed chase-based QA algorithms for (extensions of) WS Datalog$^{\pm}$
- We are working on the implementation of the QA algorithm
- We applied magic-sets techniques
- MD ontologies were motivated by data quality concerns

  They are interesting by themselves

  QA can be used to extract quality data from dirty data (RuleML'15)

- Open problems in our setting:
  - Sometimes we have to deal with closed predicates, e.g. categories
  - Inconsistency tolerance

    What if constraints are not satisfied?