Semantic Constraints for Data Quality Assessment and Cleaning

Leopoldo Bertossi
Carleton University
Ottawa, Canada

*: Faculty Fellow IBM CAS
Contents:

(A) Integrity constraints, database repairing, and CQA
(B) Data quality constraints
(C) Contexts for data quality assessment and cleaning
A. Integrity Constraints,
Database Repairing, and
Consistent Query Answering
Characterizing Consistent Data wrt ICs

A database may not satisfy a given set of integrity constraints

What is the consistent data in an inconsistent database?

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

A mathematically precise definition was needed

In (Arenas, Bertossi, Chomicki; PODS99) such a characterization was provided

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

That is, consistent data persists across all the minimally repaired versions of the original instance: the repairs of \( D \)
Example: For the instance $D$ that violates $FD: \text{Name} \rightarrow \text{Salary}$

<table>
<thead>
<tr>
<th>Employee</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>5K</td>
<td></td>
</tr>
<tr>
<td>page</td>
<td>8K</td>
<td></td>
</tr>
<tr>
<td>smith</td>
<td>3K</td>
<td></td>
</tr>
<tr>
<td>stowe</td>
<td>7K</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Employee</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>5K</td>
<td></td>
</tr>
<tr>
<td>page</td>
<td>8K</td>
<td></td>
</tr>
<tr>
<td>smith</td>
<td>3K</td>
<td></td>
</tr>
<tr>
<td>stowe</td>
<td>7K</td>
<td></td>
</tr>
</tbody>
</table>

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

(page, 8K) does not; actually it participates in the violation of $FD$
A consistent answer to a query $Q$ from a database $D$ is an answer that can be obtained as a usual answer to $Q$ from every possible repair of $D$ wrt $IC$ (a given set of ICs)

- $Q_1 : \text{Employee}(x, y)$?
  Consistent answers: $(\text{smith}, 3K), (\text{stowe}, 7K)$

- $Q_2 : \exists y \text{Employee}(x, y)$?
  Consistent answers: $(\text{page}), (\text{smith}), (\text{stowe})$

CQA may be different from classical data cleaning!

However, CQA is relevant for data quality; an increasing need in business intelligence

It also provides concepts and techniques for data cleaning
Next DBMSs should provide more flexible, powerful, and user friendlier mechanisms for dealing with semantic constraints.

In particular, they should allow to be posed queries requesting for consistent data; and answer them.

Why not an enhanced SQL?

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

(FD not maintained by the DBMS)

Paradigm shift: ICs are constraints on query answers, not on database states!
Depending on the ICs and the queries, tractable and intractable cases for CQA have been identified.

For some tractable cases, query rewriting algorithms have been developed:

\[ Q(x, y) : Employee(x, y) \mapsto Q'(x, y) : Employee(x, y) \land \neg \exists z (Employee(x, z) \land z \neq y) \]

For higher-complexity cases, specifications of repairs by means of logic programs with stable model semantics can be used:

CQA becomes querying (as usual) a logic program, say a Datalog program with possible complex extensions.
There are some implemented systems for CQA

- FO query rewriting (when possible)
- Graph-theoretic algorithmic methods
  Repairs can be implicitly represented as, e.g. maximal independent sets in a conflict graph or hypergraph
- Based on optimized (disjunctive) logic programs with stable model semantics (plus DLV)

More recently: Increasing interest in computing a single, “good” repair, or even an approximate repair

As a form of data cleaning wrt IC violation or semantic problems
A natural application: Virtual data integration

No way to enforce consistency on the sources

Inconsistencies have to be solved on-the-fly, at query time
Many problems in CQA addressed in the last few years

- Query rewriting mechanisms
- Compact representations of all DB repairs: Graph-theoretic, logic programs with stable model semantics, disjunctive databases, models of theories in non-classical logics, etc.
- Identification of tractable vs. non-tractable cases
- Applications in virtual data integration, PDMS, etc.
- Implementations

B. Data Quality Constraints
New Kinds of Constraints

Integrity constraints (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. A large body of research has been developed, in particular fundamental research. Methodologies for dealing with ICs are quite general and have broad applicability. Database repairing and CQA are newer contributions in this direction.
On the other side:

**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 data cleaning

Things are starting to change ...

Recently, DQ constraints have been proposed and investigated

They provide generic languages for expressing quality concerns

Suitable for specifying adaptive and generic DQ/C mechanisms

Proposed and studied by the Edinburgh DB group around Wenfei Fan
Conditional Dependencies (CDs)

Example: Database relation with FDs:

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

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

<table>
<thead>
<tr>
<th>CC</th>
<th>AC</th>
<th>Phone</th>
<th>Name</th>
<th>Street</th>
<th>City</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>131</td>
<td>1234567</td>
<td>mike</td>
<td>mayfield</td>
<td>NYC</td>
<td>EH4 8 LE</td>
</tr>
<tr>
<td>44</td>
<td>131</td>
<td>3456789</td>
<td>rick</td>
<td>crichton</td>
<td>NYC</td>
<td>EH4 8 LE</td>
</tr>
<tr>
<td>01</td>
<td>908</td>
<td>3456789</td>
<td>joe</td>
<td>mtn ave</td>
<td>NYC</td>
<td>07974</td>
</tr>
</tbody>
</table>

FDs are satisfied, but they are “global” ICs

They may not capture natural data quality requirements, as related to specific data values (important in data quality assessment and data cleaning)

What about a conditional functional dependency (CFD)?
\[ CFD_1 : \quad [CC = 44, Zip] \rightarrow [Street] \]

Conditional in that the FD of \textit{Street} upon \textit{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 \textit{tableau} for forced data value associations

\[ CFD_2 : \\
[CC = 44, AC = 131, Phone] \rightarrow [Street, City= 'EDI', Zip] \]

When \( CC = 44, AC = 131 \) hold, the FD of \textit{Street} and \textit{Zip} upon \textit{Phone} applies, and the city is \textit{'EDI'}

Not satisfied either ...

CQA and database repairs have been investigated for CFDs 
[Kolahi, Lakshmanan], [Beskales, Ilyas, Golab], ...
Conditional Inclusion Dependencies:

\[ \text{Order}(Title, Price, Type = 'book') \subseteq \text{Book}(Title, Price) \]

It can be expressed in classical FO predicate logic:

\[ \forall x \forall y \forall z (\text{Order}(x, y, z) \land z = 'book') \rightarrow \text{Book}(x, y) \]

Still a classic flavor ...

And semantics ...
Matching Dependencies (MDs)

MDs are related to Entity Resolution (ER)

ER is a classical, common and difficult problem in data cleaning. It is about discovering and matching records that represent the same entity in the application domain.

Again, several ad hoc mechanisms have been proposed.

ER, and DC in general, are fundamental for data analysis and decision making in BI.

Particularly crucial in data integration, and even more in virtual data integration (VDI).

In VDI, DC and ER have to be made on-the-fly, at query time.
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 $R_1(X, Y), R_2(X, Y)$

$$\forall X_1X_2Y_1Y_2(R_1[X_1] \approx R_2[X_2] \rightarrow R_1[Y_1] \doteq R_2[Y_2])$$

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 (matched)

($R_1$ and $R_2$ can be same predicate)

$\approx$: Domain-dependent similarity relation

Introduced by W. Fan et al. (PODS 2008, VLDB 2009)
Although declarative, MDs have a procedural feel and a **dynamic semantics**

An MD is satisfied by a pair of databases \((D, D')\):

- \(D\) satisfies the antecedent, and \(D'\), the consequent, where the matching is realized

But this is local, one-step satisfaction ...
Our research: [ICDT’11, LID’11]

• Alternative, refined semantics for MDs

• Investigation of the dynamic semantics

• Definition and computation of clean instances

• Definition of “clean query answering”, and computational methods to obtain them

• Comparisons between clean instances wrt MDs and database repairs wrt FDs
MDs as originally introduced do not say how to identify values

$$\forall X_1X_2Y_1Y_2(R_1[X_1] \approx R_2[X_2] \rightarrow R_1[Y_1] \doteq R_2[Y_2])$$

We have considered the two directions:

- With matching functions (MFs) (ICDT 2011), and
- Without MFs (LID 2011)
Matching Dependencies with MFs

“similar name and phone number $\Rightarrow$ identical address”

<table>
<thead>
<tr>
<th>$D_0$</th>
<th>name</th>
<th>phone</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>John Doe</td>
<td>(613)123 4567</td>
<td>Main St., Ottawa</td>
</tr>
<tr>
<td></td>
<td>J. Doe</td>
<td>123 4567</td>
<td>25 Main St.</td>
</tr>
</tbody>
</table>

⇒

<table>
<thead>
<tr>
<th>$D_1$</th>
<th>name</th>
<th>phone</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>John Doe</td>
<td>(613)123 4567</td>
<td>25 Main St., Ottawa</td>
</tr>
<tr>
<td></td>
<td>J. Doe</td>
<td>123 4567</td>
<td>25 Main St., Ottawa</td>
</tr>
</tbody>
</table>

A dynamic semantics!

$m_{address}(\text{MainSt., Ottawa}, \ 25\text{MainSt.}) := \ 25\text{MainSt., Ottawa}$

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

(Join work with Solmaz Kolahi and Laks Lakshmanan)
Semantics of MDs: [W. Fan et al., VLDB’09]

\[ \varphi : \quad R_1[X_1] \approx R_2[X_2] \rightarrow R_1[Y_1] \approx R_2[Y_2] \]

\((D, D') \models \varphi\) if for every \(R_1\)-tuple \(t_1\) and \(R_2\)-tuple \(t_2\):

\[ t_1[X_1] \approx t_2[X_2] \text{ in } D \implies t_1[Y_1] = t_2[Y_2] \text{ in } D' \]

\(D'\) is stable if \((D', D') \models \Sigma\) (a set of MDs)

Dirty instance \(D \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots \ldots \Rightarrow D'\)

\[ \uparrow \]

stable, clean instance!

• How are the MDs enforced?

• Can we expect that \((D, D') \models \Sigma\)? (too strong)
Matching Functions: Some ingredients

• Set of MDs $\Sigma$

• For every attribute $A$ with $\text{Dom}_A$
  
  – A similarity relation $\approx_A \subseteq \text{Dom}_A \times \text{Dom}_A$ reflexive and symmetric
  
  – A matching function $m_A : \text{Dom}_A \times \text{Dom}_A \rightarrow \text{Dom}_A$
    idempotent, commutative, and associative

Induces a semilattice with partial order defined as

$$a \preceq_A a' \iff m_A(a, a') = a'$$

Least upper bound operator coincides with matching function

$$\text{lub}\{a, a'\} = m_A(a, a')$$
\[ a \preceq_A a' \] can be thought of in terms of information contents

A semantic-domination lattice is created (... “domain theory”)

- **Domain-level lattice**

- **Tuple-level partial order:**
  \[ t_1 \preceq t_2 \iff t_1[A] \preceq_A t_2[A] \quad (\text{f.a. } A) \]

- **Relation-level partial order**
  \[ D_1 \sqsubseteq D_2 \iff \forall t_1 \in D_1 \ \exists t_2 \in D_2 \ t_1 \preceq t_2 \]
Instances can be “reduced” by eliminating tuples that are dominated by others

**Theorem:** The set of reduced instances with $\sqsubseteq$ forms a lattice

Relevant for comparison of sets of query answers seen as instances ...
Clean Instances:

\[ \varphi : R_1[X_1] \approx R_2[X_2] \rightarrow R_1[A_1] \cong R_2[A_2] \]

One step of chase: Enforcing \( \varphi \) on \( D \Rightarrow D' \)

- In \( D \), \( t_1[X_1] \approx t_2[X_2] \), but \( t_1[A_1] = a_1 \neq t_2[A_2] = a_2 \)
- In \( D' \), replace them with \( m_A(a_1, a_2) \)

Clean instance: Stable instance resulting from chase

\[ D_0 \Rightarrow D_1 \Rightarrow \ldots \Rightarrow D_{\text{clean}} \]

Theorem: Matching functions idem, comm, assoc give us:

(a) Chase termination after polynomial number of steps

(b) \( D_0 \sqsubseteq D_1 \sqsubseteq \ldots \sqsubseteq D_{\text{clean}} \)
In general:

- There could be multiple clean instances
- It may not hold \((D_0, D_{\text{clean}}) \models \Sigma\)

For two special cases:

- Similarity-preserving matching functions

\[ a \approx a' \implies a \approx m_A(a', a'') \]

- Interaction-free MDs

- There is a unique clean instance \(D_{\text{clean}}\), and
- \((D_0, D_{\text{clean}}) \models \Sigma\)
Clean answers to a query \( Q \): (two bounds)

- **Certain answers:** \( glb \{ Q(D) \mid D \text{ clean instance} \} \)
- **Possible answers:** \( lub \{ Q(D) \mid D \text{ clean instance} \} \)

Two clean instances:

<table>
<thead>
<tr>
<th>( D' )</th>
<th>name</th>
<th>address</th>
<th>( D'' )</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
<td>John Doe</td>
<td>Main St., Ottawa</td>
<td></td>
</tr>
<tr>
<td>J. Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
<td>J. Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
</tr>
<tr>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
</tr>
</tbody>
</table>

Query \( Q \): \( \pi_{\text{address}}(\sigma_{\text{name} = "J. Doe"}(R)) \)

**Certain** = \{25 Main St.\}

**Possible** = \{25 Main St., Ottawa, 25 Main St., Vancouver\}

**Theorem:** Computing certain clean answers is coNP-complete
Monotonicity?

$D \sqsubseteq D'$ is not set-inclusion

A query $Q$ is monotone if: $D \sqsubseteq D' \implies Q(D) \sqsubseteq Q(D')$

Why not taking advantage of lattice-theoretic domain structure when posing queries?

Proposition: A positive relational algebra query composed of $\pi$, $\times$, $\cup$, $\sigma_{a \preceq A}$, $\sigma_{A_1 \preceq A_2}$ is monotone, where

\[ t \in \sigma_{a \preceq A}(D) :\iff a \preceq t[A] \]
\[ t \in \sigma_{A_1 \preceq A_2}(D) :\iff \text{glb}\{t[A_1], t[A_2]\} \neq \perp \]

Monotonicity and clean query answering?
Two clean instances:

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>address</th>
<th></th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D'$</td>
<td>John Doe</td>
<td>25 Main St., Ottawa</td>
<td>$D''$</td>
<td>John Doe</td>
<td>Main St., Ottawa</td>
</tr>
<tr>
<td></td>
<td>J. Doe</td>
<td>25 Main St., Ottawa</td>
<td></td>
<td>J. Doe</td>
<td>25 Main St., Vancouver</td>
</tr>
<tr>
<td></td>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
<td></td>
<td>Jane Doe</td>
<td>25 Main St., Vancouver</td>
</tr>
</tbody>
</table>

Query $Q : \pi_{name} (\sigma_{25 \text{ Main St.}} \preceq address (R))$

$Q(D') = \{ \text{John Doe, J. Doe, Jane Doe} \}$

$Q(D'') = \{ \text{J. Doe, Jane Doe} \}$

Certain($Q$) = \{ J. Doe, Jane Doe \}

In this case: $Q(glb_{\subseteq} \{ D', D'' \}) = Certain(Q)$
In general, for the class $\mathcal{D}$ of clean instances

**Proposition:** For a monotone query:

\[
\begin{align*}
\underbrace{Q(\text{glb}_{\subseteq}\{D \mid D \in \mathcal{D}\})}_{\text{possible}} & \subseteq \underbrace{\text{glb}_{\subseteq}\{Q(D) \mid D \in \mathcal{D}\}}_{\text{certain}} \\
\underbrace{\text{lub}_{\subseteq}\{Q(D) \mid D \in \mathcal{D}\}}_{\text{possible}} & \subseteq \underbrace{Q(\text{lub}_{\subseteq}\{D \mid D \in \mathcal{D}\})}_{\text{certain}}
\end{align*}
\]

- Under-approximate certain answers by $Q(D_{\downarrow})$
- Over-approximate possible answers by $Q(D_{\uparrow})$

Adding heuristics to chase to obtain $D_{\downarrow}, D_{\uparrow}$?
Ongoing Research:

- Make query posing/answering sensitive to semantic-domination lattice
- Approximate query answering based on relaxation using semantic domination lattice
- Computing clean answers from data subject to MDs (without physically cleaning it)
  Query rewriting, approximations, ...
- Logic programs for clean QA in presence of MDs
  LPs specify clean instances
  LP-based declarative formulations of known ER algorithms, e.g. Swoosh
- ER and clean query answering in data integration
Declarative specifications for ER could be compiled into query answering!

On-the-fly ER!

Virtual data integration is a natural application scenario
C. Contexts for Data Quality Assessment and Data Cleaning
A table containing data about the temperatures of patients at a hospital

<table>
<thead>
<tr>
<th>Patient</th>
<th>Value</th>
<th>Time</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom Waits</td>
<td>38.5</td>
<td>11:45</td>
<td>Sep/5</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>38.2</td>
<td>12:10</td>
<td>Sep/5</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>38.1</td>
<td>11:50</td>
<td>Sep/6</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>38.0</td>
<td>12:15</td>
<td>Sep/6</td>
</tr>
<tr>
<td>Tom Waits</td>
<td>37.9</td>
<td>12:15</td>
<td>Sep/7</td>
</tr>
</tbody>
</table>

Is this quality data?

If not, is there anything to clean? What?

(Join work with Flavio Rizzolo)
We do not know ... It depends ...

Actually the table is supposed to contain *temperature measurements for Tom taken at noon by a certified nurse with an oral thermometer*

Is this quality data? We still do not know ...

Maybe we can say something about the time

Maybe good enough for the time to be “around noon” (meaning?)

Questions about the quality of this data make sense in a broader setting

The quality of the data depends on “the context”
A context that allows us to:

- make sense of the data
- assess the data
- on that basis, support data cleaning
- etc. (see below)
Contexts So Far

We find the term “context” in several places in computer science: databases, semantic web, KR, mobile applications, ...

Usually used for “context aware ... search, databases, applications, devices, ...”

Most of the time there is no explicit notion of context, but some mechanisms that take into account (or into computation) some contextual notions

Usually, time and geographic location, i.e. particular dimensions, but not much beyond
In our opinion, there is a lack of fundamental research in the area, specially for data management.

Precise and formalized notions of context are rather absent.

Contexts that can be implemented and used in a principled manner in data management systems.
Some existing research:

- Contexts in ontologies and semantic web
  Lately with emphasis on using logic programs to “bridge” implicit contexts
  Impact on data management still pending

- Contexts in KR
  They are denoted at the object level and a theory specifies their properties and dynamics
  It is possible to talk about things holding in certain (named) contexts

- Contexts in data management
  Usually in connection with specific dimensions of data, like time and place
  Relevant specific research has been carried out (Tanca et al., Torlone-Martinenghi, Spyramos et al., ...)
  A unifying framework seems to be missing
A general notion and theory of context have still to be developed. We envision it as follows:

- **A logical theory** $\mathcal{T}$ is the one that has to be “put in context.” For example, a relational database can be seen as a theory.
- **The context** is another logical theory, $\mathcal{C}$.
  $\mathcal{T}$ and $\mathcal{C}$ may share some predicate symbols.
- Actually, the **connection between $\mathcal{T}$ and $\mathcal{C}$** is established through: **connection predicates and mappings**

![Diagram](https://example.com/diagram.png)
In particular for applications in data management.

In our data quality scenario: (VLDB’10 BIRTE WS, Springer LNBIP 48, 2011)

Database $D$ can be seen as a logical theory, e.g. Reiter’s logical reconstruction of a relational DB.
In general, a contextual theory $\mathcal{C}$ and mappings and their logical/computational processing have to support what we expect from a context

- Capturing and narrowing down semantics
  - By defining in $\mathcal{C}$ predicates that are used in $\mathcal{T}$ (e.g. “time close to noon”)
  - Contributing in $\mathcal{C}$ with additional constraints for predicates used in $\mathcal{T}$, e.g. integrity constraints for table $\text{TempNoon}$
  - Term disambiguation
- Dimensions for analysis and understanding of $\mathcal{T}$’s knowledge (generalizing multidimensional DBs, DWHS)
Why not more ambitious?

- Specifying and using notions of relevance
- Explanation, diagnosis, causality
- Capturing commonsense assumptions and practices

Research has been done lately, mainly around ontologies

Has to be applied in data management

Making it accessible to “practical” DB people

There is interest in industry

- Assessment, e.g. quality
Contexts in Data Quality Assessment

- Instance $D$ is under assessment
- On RHS, also schema $S$ (or copy $S'$)
- Context $C$ is like a virtual/(semi)materialized data integration system

- The $\alpha_i$ are the mappings, like in VDISs or data exchange
- The $C_i$ are contextual predicates/relations
- There are mappings to external sources $E_i$ and quality predicates/relations $P_i$
- $D'$ 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 $C$ (mapping composition at work)

• Quality-aware (QA) query answering about (or from) $S$ can be done on top of $D'$

Techniques for query answering in VDISs can be applied (specially if $D'$ is not materialized)

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

A particular case of QA query answering
More concretely, given the data in $D$ and $C$, there may be a class $\mathcal{I}$ of admissible contextual instances $I$ for schema $C$.

Different cases, some of them ...

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

Initial table $TempNoon$ (page 37, the $R$ in $D$) is a view of $Measurements$, with mapping $\alpha$

\[
TempNoon(p, v, t, d) \leftarrow Measurements(p, v, t, d, i)
\]

Here, $\mathcal{I} = \{I\}$, a single admissible contextual instance
Now we impose quality requirements: (the $R'$ and $\alpha^P$ above)

$\text{TempNoon}'(p, v, t, d) \leftarrow \text{Measurements}(p, v, t, d, i),$

$11:30 \leq t \leq 12:30, \ i = \text{oral therm}$

Here, $R'(I) \subseteq R(D)$, and $\Delta(R(D), R'(I))$ indicates how initial $R(D)$ departs from quality instance $R'(I)$

$\text{TempNoon}'(I) \subsetneq \text{TempNoon}(D)$
Quality query answering? (conjunctive queries)

\[ Q \in L(S) \quad \Rightarrow \quad Q' \in L(S') \]

\[ R(D) \quad \Rightarrow \quad R'(I) \]

Or

\[ R'(I) \]

View unfolding:

\[ Q' \quad \Rightarrow \quad Q'' \in L(C) \quad \rightarrow \quad I \]

Here: \[ Q''(I) \subseteq Q(D) \], as expected (monotone query and additional conditions)

Here, the idea is that the database at hand is a projection of an expanded, contextual database

We work with the latter, imposing on it additional quality requirements
Example: The difference with the previous case is that we have initial instance $D$, but there is an incomplete or missing contextual instance

Here the idea is to map $D$ to the contextual schema, and impose there the quality requirements (expressed in a language associated to $\mathcal{C}$)

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

Data are in $TempNoon(D)$, no (or some) data for $Measurements$ Instrument $i$ could be obtained (or not) from additional contextual data

As in LAV: Possible several admissible instances $I$ in $\mathcal{I}$
Then, with the quality requirements:

\[ \text{TempNoon}'(p, v, t, d) \leftarrow \text{Measurements}(p, v, t, d, i), \]
\[ 11:30 \leq t \leq 12:30, \ i = \text{oral therm} \]

Possible several instances for schema \( S' \): \( D'(I) \) with \( I \in \mathcal{I} \)
\( (D'(I) \subseteq D) \)

Quality of \( D \)?

Quality measure: \( QM(D) := \frac{|D| - \max\{|D'(I)| : I \in \mathcal{I}\}}{|D|} \)

Distance to a class of quality instances (computation, estimation?)

Quality query answers?: Like certain answers on \( \{D'(I) \mid I \in \mathcal{I}\} \)
(e.g. query rewriting via rule inversion)
**Multidimensional Contexts**

Temperature data at a hospital
Doctor requires temperatures taken with oral thermometer
Doctor expects this to be reflected in the table, but the latter does not contain the information to make this assessment
An external context can provide that information, making it possible to assess the given data

The database under assessment is mapped into the context, for further data quality analysis, imposition of quality requirements, and cleaning

We can see the context as an ontology
• Hospital guideline:

“The temperature of patients in standard care units have to be taken with an oral thermometer”

Captured by means of a rule (hard, or possibly, default rule)

Or a hard constraint

• The information in the context is commonly of a multidimensional nature

We embed (an extension of) the Hurtado-Mendelzon model for MDDBs into our ontological context
A specification of the hierarchical/dimensional hospital structure

Other dimensions could be easily considered, generating multidimensional (MD) contextual information, for additional and finer-granularity data quality assessment
Contextual roll-up can be used to access missing information at certain level, by lattice navigation

Mechanisms for querying database with taxonomies could be applied/embedded (Martinenghi & Torlone; ER10)

Many interesting issues open ...
Look Ahead

The general formalization and computational use of contexts is still an open problem.

Many aspects of contexts have to be taken into account and modeled.

Ours is a long term general research.

Also in terms of applications to data quality assessment and cleaning.

We have sketched some first steps in this direction.
Next steps have to do with:

• Use of quality predicates (among those in $\mathcal{P}$ on page 44) Possibly of the kind specifically defined for capturing data quality concerns [Borgida, Mylopoulos, Lei; ER’08]

• Related to previous item, specification of sense (of data items) by imposing additional semantics

• Techniques for QA query answering
Final Remarks

In (database centered, lower-level) data management, data quality assessment usually deals with problems arising from the acquisition and integration of data: typos, inaccuracy, incompleteness, inconsistency, etc.

At the other end, BI applications require data quality assessment at higher levels of abstraction, where subjectiveness, usefulness, sense, and interpretation play a central role.

From a BI perspective, the meaning of the data, in a broad sense, and therefore its quality, are context dependent.

In our broad and long term research we are investigating the role and use of contexts in data quality assessment and cleaning.

With flexible, adaptive and generic data quality frameworks, solutions and tools in mind.