User-guided Repairing of Inconsistent Knowledge Bases

Abdallah Arioua
University of Lyon 1 / LIRIS
France

Angela Bonifati
• 2013 - 2016: PhD Computer Science (Montpellier, INRIA GraphIK team) on *Explanation and Argumentation in Inconsistent Knowledge Bases*.

• 2016 - present: Postdoc & ATER with Angela Bonifati (Lyon, LIRIS, DB team) on *Graph Databases, Cleaning and Data Integration*.

• Research interest: *Data Quality and Cleaning, Knowledge Representation, Graph Databases and Data Integration*.
Motivation
Interactive repairing of knowledge bases

- Knowledge Bases are **ubiquitous** (Semantic Web, ontology-based reasoning and data access, LOD, Big data integration and fusion, Knowledge and concept graphs in industry)
Motivation
Interactive repairing of knowledge bases

- Knowledge Bases are **ubiquitous** (Semantic Web, ontology-based reasoning and data access, LOD, Big data integration and fusion, Knowledge and concept graphs in industry)
Motivation
Interactive repairing of knowledge bases

- Errors can be introduced from mappings, typos, knowledge fusion, etc.

- Automatic repairing is *costly* and *lossy*.

- Bring human to the loop for a *better quality*. 
Preliminaries
Logical language and knowledge bases

• A knowledge base $K$ is a set of facts, TGDs (rules) and CDDs (constraints).

$$\mathcal{F} = \{\text{prescribed}(\text{Aspirin, John}), \text{hasAllergy}(\text{John, Aspirin}), \text{hasAllergy}(\text{Mike, Penicillin}), \text{hasPain}(\text{John, Migraine}), \text{isPainKillerFor}(\text{Nsaid, Migraine}), \text{incompatible}(\text{Aspirin, Nsaid})\}$$

$$\Sigma_T = \{\text{isPainKillerFor}(X, Y), \text{hasPain}(Z, Y) \rightarrow \text{prescribed}(X, Z)\}$$

$$\Sigma_C = \{\text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot, \text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot\}$$

Formalism

Tuple-generating dependency (TGD)
$$\forall x \forall y B(x, y) \rightarrow \exists z H(y, z)$$

Weakly-acyclic

Contradiction-detecting dependency (CDD)
$$\forall x B(x) \rightarrow \bot$$

Correspond to Denial Constraints with equality.
Preliminaries
Logical language and knowledge bases

• A knowledge base $K$ is a set of facts, TGDs (rules) and CDDs (constraints).

$\mathcal{F} = \{\text{prescribed(Aspirin, John), hasAllergy(John, Aspirin), hasAllergy(Mike, Penicillin), hasPain(John, Migraine), isPainKillerFor(Nsaints, Migraine), incompatible(Aspirin, Nsaints), prescribed(Nsaints, John)}\}$

$\Sigma_T = \{\text{isPainKillerFor}(X, Y), \text{hasPain}(Z, Y) \rightarrow \text{prescribed}(X, Z)\}$

$\Sigma_C = \{\text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot, \text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot\}$

Formalism

Tuple-generating dependency (TGD)
$\forall x \forall y B(x, y) \rightarrow \exists z H(y, z)$

Contradiction-detecting dependency (CDD)
$\forall x B(x) \rightarrow \bot$

Reasoning

Chase $C\ell_{\Sigma_T}(\mathcal{F}) \models Q$. 
Preliminaries
Logical language and knowledge bases

- A knowledge base $K$ is a set of facts, TGDs (rules) and CDDs (constraints).

$$ \mathcal{F} = \{ \text{prescribed(Aspirin, John)}, \text{hasAllergy(John, Aspirin)}, \text{hasAllergy(Mike, Penicillin)}, \text{hasPain(John, Migraine)}, \text{isPainKillerFor(Nsaids, Migraine)}, \text{incompatible(Aspirin, Nsaids)}, \text{prescribed(Nsaids, John)} \} $$

$$ \Sigma_T = \{ \text{isPainKillerFor}(X, Y), \text{hasPain}(Z, Y) \rightarrow \text{prescribed}(X, Z) \} $$

$$ \Sigma_C = \{ \text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot, \text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot \} $$

**Formalism**

Tuple-generating dependency (TGD)
$$ \forall x \forall y B(x, y) \rightarrow \exists z H(y, z) $$

Contradiction-detecting dependency (CDD)
$$ \forall x B(x) \rightarrow \bot $$

**Reasoning**

Chase $C\ell_{\Sigma_T}(\mathcal{F}) \models Q$. 

**Example:**

$$ Q(X) = \text{isPainKillerFor}(X, Migraine), \text{prescribed}(X, Y) $$

Ans$(Q) = \{ Nsaids \}$
Preliminaries
Inconsistency handling

• A knowledge base $K$ is inconsistent iff: $\exists N \in \Sigma_C, Cl_{\Sigma_T}(\mathcal{F}) |\models body(N)$.

Conflicts discovery

$\mathcal{F} = \{ \text{prescribed(Aspirin, John)}, \text{hasAllergy(John, Aspirin)}, \text{hasAllergy(Mike, Penicillin)}, \text{hasPain(John, Migraine)}, \text{isPainKillerFor(Nsaid, Migraine)}, \text{incompatible(Aspirin, Nsaid)}, \text{prescribed(Nsaid, John)} \}$

$\Sigma_T = \{ \text{isPainKillerFor}(X, Y), \text{hasPain}(Z, Y) \rightarrow \text{prescribed}(X, Z) \}$

$\Sigma_C = \{ \text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot, \text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot \}$

• $K$ is inconsistent: $\text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot$
  $\text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot$

• Repairing $K$ using Deletions:
  • Remove facts that are involved in conflicts.
  • Lossy approach.
Preliminaries
Inconsistency handling

• A knowledge base $K$ is inconsistent iff: $\exists N \in \Sigma_C, Cl_{\Sigma_T}(\mathcal{F}) \models body(N)$.

Conflicts discovery

$\mathcal{F} = \{\text{prescribed(Aspirin, John)}, \text{hasAllergy(John, Aspirin)}, \text{hasAllergy(Mike, Penicillin)}, \text{hasPain(John, Migraine)}, \text{isPainKillerFor(Nsaid, Migraine)}, \text{incompatible(Aspirin, Nsaid)}\}$

$\Sigma_T = \{\text{isPainKillerFor}(X, Y), \text{hasPain}(Z, Y) \rightarrow \text{prescribed}(X, Z)\}$

$\Sigma_C = \{\text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot, \text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot\}$

• $K$ is inconsistent: $\text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot$ $\text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot$

• Repairing $K$ using Updates:
  • E.g. John has an allergy against *Penicillin* rather than *Aspirin*
  • *Penicillin* is prescribed to John rather than *Aspirin*
  • ...

User interaction
Problem Statement

Given a KB equipped with a set of TGDs and CDDs, produce an error-free KB:
(i) Accounting for the interplay of TGDs and CDDs.
(ii) Minimizing user interaction.

OUTLINE

Update-based repairing
User intervention
Questioning strategies
Experimental study
Conclusion
Update-based repairing

Introduction

• Repairing using updates in RDBs\(^1\)
  • Functional Dependencies (FDs), Conditional FDs, Denial Constraints (DCs).

• Repairing using updates in KBs:
  • TGDs are natural in KB reasoning but they may introduce new conflicts.

  \[ \mathcal{F} = \{ \text{prescribed}(\text{Aspirin, John}), \text{hasAllergy}(\text{John, Aspirin}), \text{hasAllergy}(\text{Mike, Penicillin}), \text{hasPain}(\text{John, Migraine}), \text{isPainKillerFor}(\text{Nsaid,s, Migraine}), \text{incompatible}(\text{Aspirin, Nsaid},) \} \]

• Typical user interaction workflow.
Update-based repairing
Basic concepts

Position and fix
- A position is a tuple \((A, i)\) such that \(A \in \mathcal{F}\).
- A fix is triple \((A, i, t)\) such that \((A, i)\) is a position and \(t\) is a term.

- A set of fixes \(P\) is called **consistent fix** if it produces a consistent KB.
- \(P\) is a **repair fix** if KB is consistent and minimally changed (w.r.t set inclusion).
- **Example:**

\[
\mathcal{F} = \{\text{prescribed(Aspirin, John)}, \text{hasAllergy(John, Aspirin)}, \text{hasAllergy(Mike, Penicillin)}, \\
\text{hasPain(John, Migraine)}, \text{isPainKillerFor(Nsaid, Migraine)}, \text{incompatible(Aspirin, Nsaid)}\} \\
t : X_1
\]

\[
\Sigma_T = \{\text{isPainKillerFor}(X, Y), \text{hasPain}(Z, Y) \rightarrow \text{prescribed}(X, Z)\}
\]

\[
\Sigma_C = \{\text{prescribed}(X, Y), \text{hasAllergy}(Y, X) \rightarrow \bot, \\
\text{prescribed}(X, Z), \text{prescribed}(X, Y), \text{incompatible}(Y, Z) \rightarrow \bot\}
\]
Update-based repairing

\( \Pi \)-Repairability

- Repairing with immutable set of positions \( \Pi \).
- Trusted positions or previously fixed positions.
- Some KBs cannot be fixed when some positions are immutable.

**Example:**

- Consider:
  \[ \Pi = \{(p(a,b),2),(q(b,d),1)\} \]
  \[ F = \{p(a,b), q(b,d)\} \quad \Sigma_C = \{p(X,Y), q(Y,Z) \rightarrow \bot\} \]

  Not p-repairable

**Checking \( \Pi \)-repairability:**

- Change all non-immutable positions to unique labelled nulls.
- Check consistency.

  \[ F = \{p(X_1,b), q(b,X_2)\} \quad \Sigma_C = \{p(X,Y), q(Y,Z) \rightarrow \bot\} \]

  Inconsistent

- The procedure is sound, complete and computed in linear time (data complexity).
User intervention
Basic definitions

- A question $\Phi$ is a finite set of fixes.
- If all the fixes in $\Phi$ yield a $\Pi$-repairable KB then $\Phi$ is sound.
- The user chooses a fix from $\Phi$ as an answer.
- A sequence of sound questions and answers is called an inquiry over $K$.
- Example:

$$F = \{\text{prescribed(Aspirin, John)}, \text{hasAllergy(John, Aspirin)}, \text{hasAllergy(Mike, Penicillin)}, \text{hasPain(John, Migraine)}, \text{isPainKillerFor(Nsaid, Migraine)}, \text{incompatible(Aspirin, Nsaid)}\}$$

$\Phi$: which fix is true from the following set?

$$\{(\text{prescribed(Aspirin, John)}, 1, t) \mid t \in \{X_1, \text{Nsaid}\}\} \cup \{(\text{hasAllergy(John, Aspirin)}, 1, t) \mid t \in \{X_3, \text{Mike}\}\}$$

$$\{(\text{prescribed(Aspirin, John)}, 2, t) \mid t \in \{X_2, \text{Mike}\}\} \cup \{(\text{hasAllergy(John, Aspirin)}, 2, t) \mid t \in \{X_4, \text{Penicillin}\}\}$$
User intervention
Generating sound questions and inquiries

• Procedure:
  1. Generate a sound question by filtering values.
  2. Ask the user and update, continue until no conflict is left.
User intervention
Generating sound questions and inquiries

• **Procedure:**

  1. Generate a sound question by filtering values.
     1. Get all positions from the conflict
     2. Generate the proposed fixes
     3. Filter out invalid fixes using $\cap$-Repairability check.
     4. Return the question
  2. Ask the user and update, continue until no conflict is left.
User intervention
Generating sound questions and inquiries

- **Procedure:**
  1. Generate a sound question by filtering values.
     1. Get all positions from the conflict
     2. Generate the proposed fixes
     3. Filter out invalid fixes using $\prod$-Repairability check.
     4. Return the question
  2. Ask the user and update, continue until no conflict is left.
     1. Phase 1 (non-interleaving):
        1. Compute initial conflicts
        2. Generate a sound question for a given conflict
        3. Ask the user and apply the chosen fix
     2. Phase 2 (interleaving):
User intervention
Generating sound questions and inquiries

• Procedure:

1. Generate a sound question by filtering values.
   1. Get all positions from the conflict
   2. Generate the proposed fixes
   3. Filter out invalid fixes using $\prod$-Repairability check.
   4. Return the question

2. Ask the user and update, continue until no conflict is left.
   1. Phase 1 (non-interleaving):
      1. Compute initial conflicts
      2. Generate a sound question for a given conflict
      3. Ask the user and apply the chosen fix
   2. Phase 2 (interleaving):
      1. Generate sound questions when using TGDs (reasoning)
      2. Ask then apply the chosen fix
User intervention

Results

• **Proposition 1:** questions are sound and polynomial.

• **Proposition 2:** the procedure runs in finite time and produces a **consistent** knowledge base $K$.

• When the procedure produces a repair of $K$?
  • If the user is an oracle then $K$ is minimally repaired.
  • An oracle is a user who knows everything about $K$ (a domain and knowledge expert).

• **Proposition 3:** delay time between questions is **polynomial**.
Questioning strategies
Intuition by examples

- Consider the following knowledge base:

\[
\mathcal{F} = \{p(a, b, c), r(b, d), q(c, e), s(d, e), v(c), p(f, f, f)\}
\]

\[
\Sigma_C = \{p(X, Y, Z), r(Y, W), q(Z, D) \rightarrow \bot, r(X, Y), s(Y, Z) \rightarrow \bot, q(X, Y), v(X) \rightarrow \bot\}
\]

Conflicts

\[
\{p(a, b, c), r(b, d), q(c, e)\}
\]

\[
\{r(b, d), s(d, e)\}
\]

\[
\{q(c, e), v(c)\}
\]

Random strategy

1. Pick a conflict and generate all possible fixes.

\[
\phi = \{(p(a, b, c), 1, f), (p(a, b, c), 1, X_1)(p(a, b, c), 2, X_2), \ldots\}
\]

- No resolution if the user chooses: \(p(a, b, c), 1, X_1\)

- We ask more questions.

Join positions!
Questioning strategies
Intuition by examples

• Consider the following knowledge base:

\[ \mathcal{F} = \{p(a, b, c), r(b, d), q(c, e), s(d, e), v(c), p(f, f, f)\} \]

\[ \Sigma_C = \{p(X, Y, Z), r(Y, W), q(Z, D) \rightarrow \bot, \]
\[ r(X, Y), s(Y, Z) \rightarrow \bot, q(X, Y), v(X) \rightarrow \bot\} \]

Conflicts

\{p(a, b, c), r(b, d), q(c, e)\}
\{r(b, d), s(d, e)\}
\{q(c, e), v(c)\}

Join strategy

1. Pick a conflict and generate all possible fixes over join positions.

\[ \phi = \{(p(a, b, c), 2, f), (p(a, b, c), 2, X_2), (p(a, b, c), 3, X_3), (r(b, d), 1, X_4), \ldots\} \]

• We ask less questions
Questioning strategies
Intuition by examples

• Consider the following knowledge base:

\[ F = \{ p(a, b, c), r(b, d), q(c, e), s(d, e), v(c), p(f, f, f) \} \]

\[ \Sigma_C = \{ p(X, Y, Z), r(Y, W), q(Z, D) \rightarrow \bot, \\
 r(X, Y), s(Y, Z) \rightarrow \bot, q(X, Y), v(X) \rightarrow \bot \} \]

Conflicts

\{ p(a, b, c), r(b, d), q(c, e) \}
\{ r(b, d), s(d, e) \}
\{ q(c, e), v(c) \}

MCD strategy

1. Rank join positions w.r.t inclusion in conflicts and choose the top ranked.

\[ \phi = \{(q(c, e), 1, X_8)\} \]

• 2 conflicts by one question in the example.

• **Observation:** more overlapping less questions
Experimental study

Experimental environment

Variables:

- **Effectiveness**: avg number of questions per strategy and average number of conflicts per question.

- **Delay time**: average delay time between asked question.

Environment:

- Java 8, 2.40GHz 4core, 16G RAM (windows 7).
- Multi trial experiments with a cold start.
- For each experimental variable we test our approach on synthetic and real-world datasets.
Experimental study
Effectiveness

KBs:

- **Durum Wheat Kb v1**: manually constructed. TGDs and CDDs have been validated by experts.
- **Summary v1**: 567 atoms, TGDs=269, CDD=27, 185 conflicts.
- **Summary v2**: 567 atoms, TGDs=269, CDD=100, 212 conflicts.

Results:

![Graphs showing average number of questions for different strategies in Durum Wheat v1 and v2.](image-url)
Experimental study

Effectiveness: synthetic KBs no TGDs

Results:

<table>
<thead>
<tr>
<th>KB</th>
<th>Size (#atoms)</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>“05%”</td>
<td>1005</td>
<td>56</td>
</tr>
<tr>
<td>“10%”</td>
<td>1005</td>
<td>135</td>
</tr>
<tr>
<td>“16%”</td>
<td>1005</td>
<td>304</td>
</tr>
<tr>
<td>“20%”</td>
<td>1005</td>
<td>356</td>
</tr>
<tr>
<td>“25%”</td>
<td>1005</td>
<td>304</td>
</tr>
<tr>
<td>“30%”</td>
<td>1005</td>
<td>496</td>
</tr>
</tbody>
</table>
Effectiveness: synthetic KBs (convergence)

Results:

(a) Fixed size KB (3004 atoms) with constant inconsistency ratio 25%. With only CDDs and no TGDs.

(b) Fixed size KB (800 atoms) with constant inconsistency ratio of 25%, 50 CDDs and 25 TGDs. Total number of conflicts after applying the chase is 136.
Experimental study
Effectiveness: synthetic KBs (convergence)

Results:
Experimental study
Delay time: synthetic KBs only CDDs

- **Reasonable** delay time: less than 1 to 2 seconds\(^1\).
- MCD strategy is used.
- Drum wheat v1&2 less than 1 sec.

Results:

(a) Fixed size KB (3000 atoms) with increasing inconsistency ratio.

(b) Increasing size KB starting at 0% with 3000 atoms, constant inconsistency ratio 30%.

---

Experimental study
Delay time: synthetic KBs CDDs and TGDs

- **Reasonable** delay time: less than 1 to 2 seconds\(^1\).
- MCD strategy is used.
- Drum wheat v1&2 less than 1 sec.

### Results:

<table>
<thead>
<tr>
<th></th>
<th>TGDs</th>
<th>CDDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>50</td>
<td>150</td>
</tr>
<tr>
<td>D2</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>D3</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>D4</td>
<td>200</td>
<td>150</td>
</tr>
</tbody>
</table>

| Size  | 400 atoms |
| Inc ratio | 100% |

\(^1\) Robert B Miller, Response time in man-computer conversational transactions, 1968.
Conclusion

Summary:

• Update-based repairing of inconsistent knowledge bases.
• Interactive repairing in presence of interacting dependencies.
• Strategies for interaction minimization.
• Approach can be applied on portions of large knowledge bases.
• Delay time is reasonable.

Perspectives:

• Full Denial Constraints (but undecidability!).
• Other Data Cleaning constraints (CFDs, Metric FDs etc.).
End!

Thank you!
Questions

Abdallah Arioua and Angela Bonifati
“User-guided Repairing of Inconsistent Knowledge Bases”
In International Conference on Extending Database Technology (EDBT), 2018 (to appear).