Data-graph repairing: the preferred approach

Santiago Cifuentes
May 25th, 2023

Joint work with Edwin Pin, Sergio Abriola and Nina Pardal

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Department Of Computer Science
The University Of Sheffield

15th Alberto Mendelzon International Workshop
Contents

- Introduction: Data-graph repairs and Consistent Query Answering (CQA)
- Data-graphs and GXPath to express constraints
- Results
- Conclusions
What does consistency mean?

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
</tbody>
</table>

$ID \rightarrow Name$
What does \textit{consistency} mean?

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{ID} & \textbf{Name} \\
\hline
10 & Messi \\
22 & Vidal \\
09 & Suárez \\
\hline
\end{tabular}
\end{center}

\textit{ID} \rightarrow \textit{Name}
Can we *always* keep consistency?
Can we *always* keep consistency?

Not really...
Can we *always* keep consistency?

Not really...

- Systems might be poorly designed or implemented.
Can we *always* keep consistency?

Not really...

- Systems might be poorly designed or implemented.
- New requirement may arrive.
Can we *always* keep consistency?

Not really…

- Systems might be poorly designed or implemented.
- New requirement may arrive.
- It might be too expensive.
Can we *always* keep consistency?

Not really...

- Systems might be poorly designed or implemented.
- New requirement may arrive.
- It might be too expensive.

We need Inconsistency Tolerant Techniques
Repairing and Consistent Query Answering

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

$ID \rightarrow Name$
Repairing and Consistent Query Answering

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

\[ Q(name) = \exists id \; \text{Player}(id, name) \]
Repairing and Consistent Query Answering

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

Q(name) = \exists id \ Player(id, name)
### Repairing and Consistent Query Answering

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
</tbody>
</table>
Repairing and Consistent Query Answering

\[ Q(R_{messi}) \cap Q(R_{ney}) = \{ \text{Vidal, Suárez} \} \]
Repairing and Consistent Query Answering

Given a database D inconsistent with respect to a set of constraints R, a Repair of D is a new database D’ that differs minimally from D and satisfies R.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
</tbody>
</table>
Repairing and Consistent Query Answering

Given a database $D$ inconsistent with respect to a set of constraints $R$, a Repair of $D$ is a new database $D'$ that differs minimally from $D$ and satisfies $R$.

Given a query $Q$ over $D$, the Consistent Answers are defined as

$$CQA(D, R, Q) = \bigcap_{D' \in Rep(D, R)} Q(D')$$
Repairs

Different types of repairs can be defined, such as set-based or tuple-based repairs.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>
Repairs

Different types of repairs can be defined, such as set-based or tuple-based repairs.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

We will focus on set repairs.
Repairs

Set repairs must be *minimal*.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

- The first table shows the correct set of repairs.
- The second table shows an incorrect set of repairs, indicated by the red cross.
Set based Repairs

Furthermore, we will only consider **Subset** and **Superset** repairs.

- Subset repairs only allow to **remove** data.
- Superset repairs only allow to **add**.

\[ D = \{ P(a), P(b), Q(a) \} \]

\[ \forall x \ ( P(x) \rightarrow Q(x) ) \]

\[ D_{subset} = \{ P(a), Q(a) \} \]

\[ D_{superset} = \{ P(a), P(b), Q(a), Q(b) \} \]
Preferred Repairs

Also, we will study **preference criteria** among the set of repairs

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>
Preferred Repairs

Also, we will study preference criteria among the set of repairs

\[ w(R_{messi}) > w(R_{ney}) \]
Preferred Repairs

Also, we will study **preference criteria** among the set of repairs.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Messi</td>
</tr>
<tr>
<td>22</td>
<td>Vidal</td>
</tr>
<tr>
<td>09</td>
<td>Suárez</td>
</tr>
<tr>
<td>10</td>
<td>Neymar</td>
</tr>
</tbody>
</table>

\[ w(R_{messi}) > w(R_{ney}) \]
Contents

- Introduction: Data-graph repairs and Consistent Query Answering (CQA)
- Data-graphs and GXPath to express constraints
- Results
- Conclusions
Goals

❖ Define some data-aware notion of consistency for data-graphs.
Goals

❖ Define some data-aware notion of consistency for data-graphs.
❖ Study reasoning problems related to computing repairs, in particular preferred repairs.
Goals

❖ Define some data-aware notion of consistency for data-graphs.
❖ Study reasoning problems related to computing repairs, in particular preferred repairs.
❖ Find tractable cases of the problems.
The Model

We will consider RDF graphs.
The Constraints

Other works have already developed constraints over data-graphs

\[
\begin{align*}
\textit{RPCs} & \quad \{ \hspace{1cm} \text{child_of} \subseteq \text{son_of} \cup \text{daughter_of}. \\
& \hspace{1cm} \text{child_of} \cdot (\text{brother_of} \cup \text{sister_of}) \subseteq \text{nephew_of} \cup \text{niece_of}. \}
\end{align*}
\]

Barceló & Fontaine, 2017
The Constraints

Other works have already developed constraints over data-graphs

\[
\begin{align*}
\text{RPCs} \quad & \quad \text{child_of} \subseteq \text{son_of} \cup \text{daughter_of}. \\
& \quad \text{child_of} \cdot (\text{brother_of} \cup \text{sister_of}) \subseteq \text{nephew_of} \cup \text{niece_of}.
\end{align*}
\]

Based on Regular Path Expressions

Barceló & Fontaine, 2017
The Constraints

Other works have already developed constraints over data-graphs

\[
\begin{align*}
\text{RPCs} & \quad \begin{cases}
\text{child_of} \subseteq \text{son_of} \cup \text{daughter_of}.
\text{child_of} \cdot (\text{brother_of} \cup \text{sister_of}) \subseteq \text{nephew_of} \cup \text{niece_of}.
\end{cases}
\end{align*}
\]

Based on Regular Path Expressions, only considers the topology

Barceló & Fontaine, 2017
The Constraints

We might also want to ask for *nested paths*
The Constraints

We might also want to ask for nested paths

Moreover, we want to interact with data

Perez & all, 2010
GXPath

Language that extends common RPQs, NREs and allows to interact with data through equality tests and data comparison

$$\varphi, \psi := c \|= c \neq | \neg \varphi | \varphi \lor \psi | \varphi \land \psi | \langle \alpha \rangle | \langle \alpha = \beta \rangle | \langle \alpha \neq \beta \rangle$$

$$\alpha, \beta := \epsilon | \_ | \mathbb{A} | \mathbb{A}^- | \alpha \circ \beta | \alpha \cup \beta | \alpha \cap \beta | \alpha^* | \overline{\alpha} | \varphi | \alpha^{n,m}$$
GXPath

Language that extends common RPQs, NREs and allows to interact with data through equality tests and data comparison

\[ \langle \downarrow \text{acts\_in} \ [\text{Babylon=}] \rangle \]
GXPath

Language that extends common RPQs, NREs and allows to interact with data through equality tests and **data comparison**

$$\langle \downarrow_{\text{type}} [\text{Film}] \rangle \land \langle \downarrow_{\text{acts\_in}} \neq \downarrow_{\text{acts\_in}} \rangle$$

Libkin & all, 2016
GXPath

Language that extends common RPQs, NREs and allows to interact with data through equality tests and data comparison.

It can be evaluated in polynomial time

Libkin & all, 2016
GXPath as a constraint language

Given a data-graph G and an expression E, we will say that G is consistent w.r.t E is E captures all nodes from G
GXPath as a constraint language

Given a data-graph G and an expression E, we will say that G is consistent w.r.t E is E captures all nodes from G

\[ \left\langle \downarrow\text{child\_of} \cup \downarrow\text{sibling\_of} \right\rangle \]
GXPath as a constraint language

Given a data-graph $G$ and an expression $E$, we will say that $G$ is consistent w.r.t. $E$ if $E$ captures all nodes from $G$. 

\[
\xymatrix{
\text{Diego} & \text{Julieta} \\
\text{María} & \text{Mauro} \\
\text{sibling}_\text{of} & \text{sibling}_\text{of} & \text{sibling}_\text{of} & \text{sibling}_\text{of}
}
\]
GXPath as a constraint language

Given a data-graph $G$ and an expression $E$, we will say that $G$ is consistent w.r.t $E$ if $E$ captures all nodes from $G$. 

![Diagram](image-url)
GXPath as a constraint language

Given a data-graph $G$ and an expression $E$, we will say that $G$ is consistent w.r.t $E$ if $E$ captures all nodes from $G$.
GXPath as a constraint language

\[
\langle \downarrow_{\text{country}} [\text{France}=] \rangle \rightarrow \langle (\downarrow_{\text{edge}} (\downarrow_{\text{sp}} [\text{transport}=]) \downarrow_{\text{node}})^* [\text{Hastings}=] \rangle
\]
Preference criteria

We will study two preference criteria. The first one is based on a weight function:

\[ w : \Sigma_e \sqcup \Sigma_n \rightarrow \mathbb{N} \]

\[ w(G) = \sum_{x, y \in V} \left( \sum_{z \in L_e(x, y)} w(z) \right) + \sum_{x \in V} w(D(x)) \]
Weight-based preference

\[ \alpha = \downarrow^* \]

\[ \beta = (\downarrow_{\text{high}} \cup \downarrow_{\text{low}} \downarrow_{\text{high}})^* \cdot (\downarrow_{\text{low}} \cup \epsilon) \]

\[ R = \{ \alpha, \beta \} \]

\[ w(\text{low}) = 1 \]

\[ w(\text{high}) = 5 \]
Weight-based preference

\[ \alpha = \downarrow^* \]

\[ \beta = (\downarrow_{\text{high}} \cup \downarrow_{\text{low}} \downarrow_{\text{high}})^* \cdot (\downarrow_{\text{low}} \cup \epsilon) \]

\[ R = \{ \alpha, \beta \} \]

\[ w(\text{low}) = 1 \]

\[ w(\text{high}) = 5 \]
Preference criteria

We second one is based on lifting an order from the elements of the graph to the graphs themselves:

\[ G^M = \bigcup_{x,y \in V} L_e(x, y) \cup \bigcup_{x \in V} D(x) \]

\[ M_1 \lessdot_{\text{mset}} M_2 \text{ iff } M_1 \neq M_2 \text{ and for all } x \in A, \text{ if } M_1(x) > M_2(x), \text{ then there exists some } y \in A \text{ such that } x < y \text{ and } M_1(y) < M_2(y). \]
Order-based preference

\[ \alpha = \epsilon \rightarrow \underline{+} \]

\[ \phi = C^= \rightarrow (\langle \downarrow [R^=] \rangle \vee (\langle \downarrow [D^=] \rangle^{1000,1000})) \]

\[ R = \{ \alpha, \phi \} \]

\[ \downarrow < C < D < R \]
Order-based preference

\[ \alpha = \epsilon \rightarrow \bar{+} \]

\[ \phi = C^= \rightarrow (\downarrow [R^=]) \lor ((\downarrow [D^=])^{1000,1000}) \]

\[ R = \{\alpha, \phi\} \]

\[ \downarrow < C < D < R \]
The problem

**PROBLEM:** $\Omega$-PREFERRED $\star$-REPAIR

**INPUT:** A data-graph $G$ and a set $R$ of expressions from $\mathcal{L}$.

**OUTPUT:** Find a $\Omega$-preferred $\star$-repair of $G$ with respect to $R$. 
The problem

**Problem:** $\Omega$-preferred $\ast$-repair  
**Input:** A data-graph $G$ and a set $R$ of expressions from $\mathcal{L}$.  
**Output:** Find a $\Omega$-preferred $\ast$-repair of $G$ with respect to $R$.  

The languages we will consider for the expressions are:
The problem

**Problem:** Ω-preferred ∗-repair
**Input:** A data-graph $G$ and a set $R$ of expressions from $L$.
**Output:** Find a Ω-preferred ∗-repair of $G$ with respect to $R$.

The languages we will consider for the expressions are:

- Full GXPath.
The problem

**Problem:** Ω-preferred ★-repair

**Input:** A data-graph $G$ and a set $R$ of expressions from $\mathcal{L}$.

**Output:** Find a Ω-preferred ★-repair of $G$ with respect to $R$.

The languages we will consider for the expressions are:

- Full GXPath.
- Positive GXPath (i.e. without negation)
The problem

**Problem:** $\Omega$-preferred $\ast$-repair

**Input:** A data-graph $G$ and a set $R$ of expressions from $\mathcal{L}$.

**Output:** Find a $\Omega$-preferred $\ast$-repair of $G$ with respect to $R$.

The languages we will consider for the expressions are:

- Full GXPath.
- Positive GXPath (i.e. without negation)
- Only the node / path fragment.
Contents

- Introduction: Data-graph repairs and Consistent Query Answering (CQA)
- Data-graphs and GXPath to express constraints
- Results
- Conclusions
### Previous work

<table>
<thead>
<tr>
<th>Repair kind</th>
<th>Expressions</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset</td>
<td>Positive path GXPath</td>
<td>NP-completo</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
<tr>
<td>Superset</td>
<td>GXPath</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Positive GXPath</td>
<td>PTIME in data complexity</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
</tbody>
</table>

Abriola & all, 2023
**Previous work**

<table>
<thead>
<tr>
<th>Repair kind</th>
<th>Expressions</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset</td>
<td>Positive path GXPath</td>
<td>NP-completo</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
<tr>
<td>Superset</td>
<td>GXPath</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Positive GXPath</td>
<td>PTIME in data complexity</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
</tbody>
</table>

*Abriola & all, 2023*
Subset preferred repairs

**Corollary 14.** There exists an algorithm that given a data-graph $G$, a set of Reg-GXPath$^{pos}_\text{node}$ expressions $R$, and a preference criteria $\Omega$ computes a $\Omega$-preferred $\subseteq$-repair of $G$ with respect to $R$ in $O(n \times \text{eval}(R))$. 
Subset preferred repairs

**Corollary 14.** There exists an algorithm that given a data-graph $G$, a set of Reg-GXPath$_{\text{node}}^{\text{pos}}$ expressions $R$, and a preference criteria $\Omega$ computes a $\Omega$-preferred $\subseteq$-repair of $G$ with respect to $R$ in $O(n \times \text{eval}(R))$.

**Theorem 15.** Given a data-graph $G$ and a set of Reg-GXPath$_{\text{node}}^{\text{pos}}$ node expressions $R$ there is an unique subset repair of $G$ with respect to $R$
**Algorithm 1** \textit{SubsetRepair}(G, R)

\textbf{Require:} G is a data-graph and R a set of Reg-GXPath\textsuperscript{pos} node expressions.

1: while (G, R) is inconsistent do
2: \quad V_\bot \leftarrow \{ v \mid v \in V_G \text{ and } \exists \varphi \in R \text{ such that } v \notin [\varphi]_G \}
3: \quad G \leftarrow G_{V_G \setminus V_\bot}
4: end while
5: return \ G

Abriola & all, 2023
Superset preferred repairs

There were two tractable scenarios:
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are positive and fixed
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are positive and fixed

Observation 23. If there is a superset repair $G$ with respect to $R$, then there exists a superset repair with a number of different data values that linearly depends on $|G| + |R|$.

Abriola & all, 2023
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are **positive** and **fixed**

**Observation 23.** *If there is a superset repair G with respect to R, then there exists a superset repair with a number of different data values that linearly depends on |G| + |R|.*

Abriola & all, 2023

We can show that this “standard“ repair is preferred for both criteria
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are positive and fixed

observation 23. If there is a superset repair $G$ with respect to $R$, then there exists a superset repair with a number of different data values that linearly depends on $|G| + |R|$.

Abriola & all, 2023

Therefore, we can compute a preferred repair for both criteria in PTIME for data complexity
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are **positive** and **fixed**
2. When we only consider **positive node expressions**
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are **positive** and **fixed**
2. When we only consider **positive node expressions**

**Theorem 15.** Given a data-graph $G$, a set of \( \text{Reg-GXPath}_{\text{node}}^{\text{pos}} \) expressions $R$ and a natural number $K$, let $\Pi_w$ be the problem of deciding whether there exists a $w$-preferred $\supseteq$-repair of $G$ with respect to $R$ whose weight is bounded by $K$. Then, there exists a set of positive node expressions $R$ and a weight function $w$ such that the problem is \text{NP-COMPLETE}.
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are **positive** and **fixed**
2. When we only consider **positive node expressions**

**Theorem 15.** Given a data-graph $G$, a set of $\text{Reg-GXPath}^{\text{pos}}_{\text{node}}$ expressions $R$ and a natural number $K$, let $\Pi_w$ be the problem of deciding whether there exists a $w$-preferred $\supseteq$-repair of $G$ with respect to $R$ whose weight is bounded by $K$. Then, there exists a set of positive node expressions $\mathcal{R}$ and a weight function $w$ such that the problem is $\text{NP}$-complete.

Intractable for weight-based preference
Superset preferred repairs

There were two tractable scenarios:

1. When the expressions are **positive** and **fixed**
2. When we only consider **positive node expressions**

**Theorem 15.** Given a data-graph $G$, a set of Reg-GXPath_{node}^{pos}$ expressions $R$ and a natural number $K$, let $\Pi_w$ be the problem of deciding whether there exists a $w$-preferred $\succeq$-repair of $G$ with respect to $R$ whose weight is bounded by $K$. Then, there exists a set of positive node expressions $R$ and a weight function $w$ such that the problem is NP-COMPLETE.

Intractable for weight-based preference

Similar for proof for order based preference
Pushing a bit more

Inspecting the intractability proof for weight based it is clearly seen that it depends on assigning weights to the edges. What happens if we forbid this?
Pushing a bit more

Inspecting the intractability proof for weight based it is clearly seen that it depends on assigning weights to the edges. What happens if we forbid this?

**Theorem 18.** There exists an algorithm that given a data-graph $G$ and a set of $\text{Reg-GXPath}_{\text{node}}^{\text{pos}}$ expressions $R$ computes a $w$-preferred $\sqsupset$-repair in polynomial time in data complexity whenever $w$ satisfies that $w(A) = 0$ for all $A \in \Sigma_e$. 
Pushing a bit more

Inspecting the intractability proof for weight based it is clearly seen that it depends on assigning weights to the edges. What happens if we forbid this?

**Theorem 18.** There exists an algorithm that given a data-graph $G$ and a set of $\text{Reg-GXPath}^{\text{pos}}_{\text{node}}$ expressions $R$ computes a $w$-preferred $\supseteq$-repair in polynomial time in data complexity whenever $w$ satisfies that $w(A) = 0$ for all $A \in \Sigma_e$.

We recover tractability, but only in data complexity
Pushing a bit more

**Theorem 19.** There is a fixed weight function \( w \) that assigns 0 cost to all edge labels such that the problem \( \Pi_w \) (see Theorem 15) is NP-COMPLETE in combined complexity when restricting that \( R \subseteq \text{Reg-GXPath}^{pos}_{node} \).

It is still hard in combined complexity
## Previous work

<table>
<thead>
<tr>
<th>Repair kind</th>
<th>Expressions</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset</td>
<td>Positive path GXPath</td>
<td>NP-completo</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
<tr>
<td>Superset</td>
<td>GXPath</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Positive GXPath</td>
<td>PTIME in data complexity</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
</tbody>
</table>
## With preferences

<table>
<thead>
<tr>
<th>Repair kind</th>
<th>Expressions</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subset</strong></td>
<td>Positive path GXPath</td>
<td>NP-complete</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
<tr>
<td><strong>Superset</strong></td>
<td>GXPath</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Positive GXPath</td>
<td>PTIME in data complexity</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td><strong>PTIME NP-complete</strong></td>
</tr>
</tbody>
</table>

Abriola & all, 2023
<table>
<thead>
<tr>
<th>Repair kind</th>
<th>Expressions</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset</td>
<td>Positive path GXPath</td>
<td>NP-completo</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME</td>
</tr>
<tr>
<td>Superset</td>
<td>Positive path GXPath</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME in data</td>
</tr>
<tr>
<td></td>
<td>Positive node GXPath</td>
<td>PTIME NP-complete</td>
</tr>
</tbody>
</table>

Abriola & all, 2023
Conclusions

❖ We proposed a consistency definition based on GXPath able to express some good properties of data-graphs.
Conclusions

❖ We proposed a consistency definition based on GXPath able to express some good properties of data-graphs.
❖ We gave two preference criteria such that the problem of computing preferred repairs is as hard as computing “common” repairs, in most cases.
Conclusions

❖ We proposed a consistency definition based on GXPath able to express some good properties of data-graphs.
❖ We gave two preference criteria such that the problem of computing preferred repairs is as hard as computing “common” repairs, in most cases.
❖ For the cases that became intractable some conditions were found that recover tractability (at least in data complexity).
Future work

- Study other kind of repairs, such as those based on symmetric difference or updating data values.
Future work

- Study other kind of repairs, such as those based on symmetric difference or updating data values.
- Study CQA in this context.
Future work

- Study other kind of repairs, such as those based on symmetric difference or updating data values.
- Study CQA in this context.
- Try to include some form of negation (in particular, in node expressions).