Schema Mappings for Data Graphs

Nadime Francis\textsuperscript{1} Leonid Libkin\textsuperscript{2}

\textsuperscript{1}Université Paris-Est Marne-la-Vallée
\textsuperscript{2}University of Edinburgh

BDA 2017
Nancy, November, 16th

Results originally published in PODS 2017
INTRODUCTION
Data Exchange

Legacy Database over $\Sigma_s$ \quad \rightarrow \quad \mathcal{M} \quad \rightarrow \quad \{ \text{Possible Solutions over } \Sigma_t \}

Schema Mapping from $\Sigma_s$ to $\Sigma_t$
Data Exchange

Legacy Database over $\Sigma_s$ → $\mathcal{M}$ → Schema Mapping from $\Sigma_s$ to $\Sigma_t$ → Possible Solutions over $\Sigma_t$

Query over $\Sigma_t$ → Certain answers
Data Exchange as Virtual Data Integration

Source Databases over $\Sigma_s = \{\Sigma_1, \Sigma_2, \Sigma_3\}$

Schema Mapping from $\Sigma_s$ to $\Sigma_t$

Virtual Database over mediated schema $\Sigma_t$

Query over $\Sigma_t$ 

Integrated answers
Key questions

Solution Existence

- Input: A database $S$ and a mapping $\mathcal{M}$
- Question: Does there exist a solution $T$ to $S$ under $\mathcal{M}$?
Key questions

Solution Existence
- Input: A database $S$ and a mapping $\mathcal{M}$
- Question: Does there exist a solution $T$ to $S$ under $\mathcal{M}$?

Query Answering
- Input: A database $S$, a tuple $\bar{v} \in S$, a mapping $\mathcal{M}$, a query $Q$
- Question: Is $\bar{v}$ a “certain answer” to $Q$ over the solution to $S$ under $\mathcal{M}$?
Why look at graph data?

- Data naturally presented as graphs:
  → Social networks, Internet, biological data, Semantic Web...
- Queries care about the topology of the graph:
  → Paths, cycles, connected components, graph patterns...
- Known techniques for relational databases do not apply
Property Graphs, Graph Databases and Data Graphs

- **Property graphs**: real life model, used in Neo4j (Cypher), advocated by LDBC.

- Graph databases: theoretical modelization as edge-labelled graph. **Topology only**. Does not capture real life scenarios.

- Data graphs: edge-labelled graphs with nodes carrying values from an infinite domain. Better abstraction of real life cases. **Topology and data**.

![Diagram](attachment://diagram.png)
Property Graphs, Graph Databases and Data Graphs

- **Property graphs**: real life model, used in Neo4j (Cypher), advocated by LDBC.

- **Graph databases**: theoretical modelization as edge-labelled graph. **Topology only**. Does not capture real life scenarios.

- **Data graphs**: edge-labelled graphs with nodes carrying values from an infinite domain. Better abstraction of real life cases. **Toplogy and data**.
Property Graphs, Graph Databases and Data Graphs

- **Property graphs**: real life model, used in Neo4j (Cypher), advocated by LDBC.

- **Graph databases**: theoretical modelization as edge-labelled graph. **Topology only**. Does not capture real life scenarios.

- **Data graphs**: edge-labelled graphs with nodes carrying values from an infinite domain. Better abstraction of real life cases. **Topology and data**.
DATA EXCHANGE AND QUERY ANSWERING
Known Results

Data Exchange for Relational Databases

- Extensively investigated over the last 20 years
- “Book material”:
  - Foundations of Data Exchange [Arenas, Barceló, Libkin, Murlak, 2014]
  - Principles of Data Integration [Doan, Halevy, Ives, 2012]
- When and how solutions can be efficiently built and queried
Known Results

Data Exchange for Relational Databases
- Extensively investigated over the last 20 years
- "Book material":
  - Foundations of Data Exchange [Arenas, Barceló, Libkin, Murlak, 2014]
  - Principles of Data Integration [Doan, Halevy, Ives, 2012]
- When and how solutions can be efficiently built and queried

Data Exchange for Graph Data
- Schema Mappings and Data Exchange for Graph Databases [Barceló, Pérez, Reutter, 2013]
- Answering queries is typically coNP, tractability requires RPQs in mappings to be "rigid" (no *, no ∨)
Known Results

Data Exchange for Relational Databases
- Extensively investigated over the last 20 years
- “Book material”:
  - Foundations of Data Exchange [Arenas, Barceló, Libkin, Murlak, 2014]
  - Principles of Data Integration [Doan, Halevy, Ives, 2012]
- When and how solutions can be efficiently built and queried

Data Exchange for Graph Data
- Schema Mappings and Data Exchange for Graph Databases [Barceló, Pérez, Reutter, 2013]
- Answering queries is typically coNP, tractability requires RPQs in mappings to be “rigid” (no *, no ∨)

→ No literature for data graphs!
### Chosen Setting

#### Key parameters

- **Source** ($G_s$) and target ($G_t$) databases: Data graphs
- Schema mapping ($\mathcal{M}$): Pairs of RPQs
- Query ($Q$): Data RPQs
Chosen Setting

Key parameters

- Source \((G_s)\) and target \((G_t)\) databases: **Data graphs**
- Schema mapping \((\mathcal{M})\): Pairs of RPQs
- Query \((Q)\): Data RPQs

- Source: data graph over \(\Sigma_s\) and \(D\)
- Target: data graph over \(\Sigma_t\) and \(D\)
Chosen Setting

Key parameters

- Source \((G_s)\) and target \((G_t)\) databases: Data graphs
- Schema mapping \((\mathcal{M})\): **Pairs of RPQs**
- Query \((Q)\): Data RPQs

\[
\mathcal{M} = \{(q_i, q'_i)\} \quad (G_s, G_t) \models \mathcal{M} \iff q_i(G_s) \subseteq q'_i(G_t)
\]

Note: nodes of \(G_s\) are exported together with their data values

Ex: \(\mathcal{M} = \{(ba^+, c); (ac, ab^*)\}\)
**Chosen Setting**

**Key parameters**

- Source ($G_s$) and target ($G_t$) databases: Data graphs
- Schema mapping ($\mathcal{M}$): **Pairs of RPQs**
- Query ($Q$): Data RPQs

$$\mathcal{M} = \{(q_i, q'_i)\} \quad (G_s, G_t) \models \mathcal{M} \iff q_i(G_s) \subseteq q'_i(G_t)$$

Note: nodes of $G_s$ are exported together with their data values

Ex: $\mathcal{M} = \{(ba^+, c); (ac, ab^*)\}$
Chosen Setting

Key parameters

- Source \((G_s)\) and target \((G_t)\) databases: Data graphs
- Schema mapping \((\mathcal{M})\): **Pairs of RPQs**
- Query \((Q)\): Data RPQs

\[
\mathcal{M} = \{(q_i, q'_i)\} \quad (G_s, G_t) \models \mathcal{M} \text{ iff } q_i(G_s) \subseteq q'_i(G_t)
\]

Note: nodes of \(G_s\) are exported **together with their data values**

Ex: \(\mathcal{M} = \{(ba^+, c); (ac, ab^*)\}\)
Chosen Setting

Key parameters

- Source \((G_s)\) and target \((G_t)\) databases: Data graphs
- Schema mapping \((\mathcal{M})\): Pairs of RPQs
- Query \((Q)\): Data RPQs

Most general: regular expressions with memory

Ex:

\[
Q = \Sigma^* \cdot \downarrow x. (\Sigma^+_t[x=]) \cdot (\Sigma^+_t[x=]) \cdot \Sigma^*_t \cdot (a \mid b)
\]

A data value repeats three times along a path finishing with \(a\) or \(b\).

\[
Q(G_t) = \{ (\bullet, \bullet); (\bullet, \bullet); (\bullet, \bullet); (\bullet, \bullet) \}
\]
Query Answering: Certain Answers Semantics

Certain Answers

Let $G_s$ be a source database, $\mathcal{M}$ be a schema mapping from $\Sigma_s$ to $\Sigma_t$, and $Q$ be a query over $\Sigma_t$. Then:

$$\square_\mathcal{M}(Q, G_s) = \bigcap_{G_t \mid (G_s, G_t) \models \mathcal{M}} Q(G_t)$$
Query Answering: Certain Answers Semantics

Certain Answers

Let $G_s$ be a source database, $\mathcal{M}$ be a schema mapping from $\Sigma_s$ to $\Sigma_t$, and $Q$ be a query over $\Sigma_t$. Then:

$$\square_{\mathcal{M}}(Q, G_s) = \bigcap_{G_t \mid (G_s, G_t) \models \mathcal{M}} Q(G_t)$$

Problem: Query Answering

Input: A data graph $G_s$, a tuple $\vec{v}$ of nodes of $G_s$, a schema mapping $\mathcal{M}$ and a query $Q$

Question: Is $\vec{v}$ in $\square_{\mathcal{M}}(Q, G_s)$?
Query Answering: Certain Answers Semantics

Certain Answers

Let $G_s$ be a source database, $\mathcal{M}$ be a schema mapping from $\Sigma_s$ to $\Sigma_t$, and $Q$ be a query over $\Sigma_t$. Then:

$$\square_\mathcal{M}(Q, G_s) = \bigcap_{G_t \mid (G_s, G_t) \models \mathcal{M}} Q(G_t)$$

**Problem**: Query Answering $(\mathcal{M}, Q)$ Data complexity

**Input**: A data graph $G_s$, a tuple $\bar{v}$ of nodes of $G_s$

a schema mapping $\mathcal{M}$ and a query $Q$

**Question**: Is $\bar{v}$ in $\square_\mathcal{M}(Q, G_s)$?
Contributions

Undecidability and Intractability

- Query answering is **undecidable** for RPQ mappings and data RPQ queries. (Already true for very simple mappings)
- Query answering is **coNP-complete** for word mappings and data RPQ queries. (Already true for paths with tests)

Recovering Tractability

- Query answering is in **NLogSpace** for word mappings and data RPQ queries under the presence of SQL nulls.
- Query answering is in **NLogSpace** for word mappings and data RPQ queries with equality only.
Undecidability and Intractability
Undecidability

**Theorem**

There exist a very simple mapping $\mathcal{M}$ and an RPQ with equality $Q$ such that $\text{QUERY ANSWERING} (\mathcal{M}, Q)$ is undecidable.
Theorem

There exist a very simple mapping $\mathcal{M}$ and an RPQ with equality $Q$ such that $\text{QUERY ANSWERING}(\mathcal{M}, Q)$ is undecidable.

Very simple:
Rules of $\mathcal{M}$ are of two possible forms:
- Either $(a, b)$, with $a \in \Sigma_s$ and $b \in \Sigma_t$;
- Or $(a, \Sigma_t^*)$, with $a \in \Sigma_s$. 
Undecidability

Theorem

There exist a very simple mapping \( M \) and an RPQ with equality \( Q \) such that \( \text{QUERY ANSWERING}(M, Q) \) is undecidable.

RPQ with equality:

- Query defined by an expression with \( e_\equiv \) and \( e_\neq \) subexpressions
  
  Ex: \( \Sigma^* \cdot (a^+) \equiv \cdot \Sigma^* \)
  
  A data value occurs twice with only \( a \)'s in between.

- Strictly weaker than data RPQs.
Theorem

There exist a very simple mapping $M$ and an RPQ with equality $Q$ such that QUERY ANSWERING$(M, Q)$ is undecidable.

For these fixed (data complexity) $M$ and $Q$, given:

- a database $G_s$ over $\Sigma_s$,
- a pair of nodes $(x, y)$ of $G_s$,

it is undecidable whether $(x, y) \in \Box_M(Q, G_s)$. 
Intractability

**Theorem**

Let $\mathcal{M}$ be a relational mapping and a $Q$ a data RPQ. Then $\text{QUERY ANSWERING}(\mathcal{M}, Q)$ is coNP-complete.
Theorem

Let $\mathcal{M}$ be a relational mapping and a $Q$ a data RPQ. Then $\text{QUERY ANSWERING}(\mathcal{M}, Q)$ is coNP-complete.

Relational mapping:

Rules of $\mathcal{M}$ are of the form $(q, w)$, where:

- $q$ is an arbitrary RPQ over $\Sigma_s$;
- $w$ is a single word over $\Sigma_t$.

It is already coNP-hard when $q$ is a single symbol $a \in \Sigma_s$. 
Intractability

Theorem

Let $M$ be a relational mapping and a $Q$ a data RPQ. Then $\text{QUERY ANSWERING}(M, Q)$ is coNP-complete.

Data RPQ:

- Any query defined by a register automaton or a regular expression with memory.
- It is already hard if $Q$ is a path with tests. Ex: $((ab)c) \neq$. No disjunction, no transitive closure.
Theorem

Let $\mathcal{M}$ be a relational mapping and a $Q$ a data RPQ. Then $\text{QUERY ANSWERING}(\mathcal{M}, Q)$ is coNP-complete.

For these fixed (data complexity) $\mathcal{M}$ and $Q$, given:

- a database $G_s$ over $\Sigma_s$,
- a pair of nodes $(x, y)$ of $G_s$,

it is coNP-complete to decide whether $(x, y) \in \square_{\mathcal{M}}(Q, G_s)$. 
TRACTABLE FRAGMENTS
Introducing SQL Nulls

SQL Behavior

- SQL: Missing values modelled with a single null value $\mathbf{N}$.
- Three valued logic:
  - $x = y$ returns $unknown$ if either $x = \mathbf{N}$ or $y = \mathbf{N}$;
  - Otherwise, $x = y$ evaluates as usual.
- At query answering time, $unknown$ collapses to $false$. 
Introducing SQL Nulls

SQL Behavior

- SQL: Missing values modelled with a single null value $\text{N}$.
- Three valued logic:
  - $x = y$ returns $unknown$ if either $x = \text{N}$ or $y = \text{N}$;
  - Otherwise, $x = y$ evaluates as usual.
- At query answering time, $unknown$ collapses to $false$.

Translating to data RPQs

- Replace $D$ with $D_N = D \cup \{\text{N}\}$.
- Comparisons: as in SQL.
Query Answering Approximation via SQL Nulls

Theorem

Let $\mathcal{M}$ be a relational mapping and $Q$ be a data RPQ. Then, given a source database $G_s$ and $(x, y)$ in $G_s$, it can be decided in NLogSpace whether $(x, y) \in \square^N_{\mathcal{M}}(Q, G_s)$.
Query Answering Approximation via SQL Nulls

**Theorem**

Let $\mathcal{M}$ be a relational mapping and $Q$ be a data RPQ. Then, given a source database $G_s$ and $(x, y)$ in $G_s$, it can be decided in NLogSpace whether $(x, y) \in \square^n_{\mathcal{M}}(Q, G_s)$.

\[ \square^n_{\mathcal{M}}(Q, G_s) = \bigcap_{G_t \text{ over } \mathcal{D}_N} Q(G_t) \]

Motivations

$\square^n_{\mathcal{M}}(Q, G_s)$ is an underapproximation of $\square_{\mathcal{M}}(Q, G_s)$.

It matches the implementation of traditional DBMSs, and thus the expected behavior if $G_t$ is materialized as such.
Query Answering Approximation via SQL Nulls

Theorem

Let $\mathcal{M}$ be a relational mapping and $Q$ be a data RPQ. Then, given a source database $G_s$ and $(x, y)$ in $G_s$, it can be decided in NLogSpace whether $(x, y) \in \Box^N_{\mathcal{M}}(Q, G_s)$.

$$\Box^N_{\mathcal{M}}(Q, G_s) = \bigcap_{G_t \text{ over } \mathcal{D}_N} Q(G_t)$$

Motivations

- $\Box^N_{\mathcal{M}}(Q, G_s)$ is an underapproximation of $\Box_{\mathcal{M}}(Q, G_s)$.
- It matches the implementation of traditional DBMSs, and thus the expected behavior if $G_t$ is materialized as such.
Queries without Inequalities

Data RPQs without Inequalities

- Expressions with memory: only \([x = \_]\).
- Expressions with equality: only \(e_\leq\).
### Queries without Inequalities

#### Data RPQs without Inequalities
- Expressions with memory: only $[x =]$.  
- Expressions with equality: only $e = $.

#### Results
- `QUERY ANSWERING(M, Q)` is in $\text{NLogSpace}$ for relational mappings and data RPQs without inequalities.  
- `QUERY ANSWERING(M, Q)` is in $\text{coNP}$ for RPQ mappings and RPQ with equalities and no inequalities.  
- The question remains open when $Q$ is an RPQ with memory and no inequalities.
PERSPECTIVES
Very first steps in Data Exchange for Data Graphs
- Early picture of decidability / tractability
- Other tasks (metadata management)

Application to real life graph data
- Property graphs and real-life query languages
- Neo Technology: Neo4j and Cypher

Back to relational data exchange
- Use of SQL nulls instead of marked nulls
- Get efficient approximations
Thank you!