Schema Mappings and Data Exchange for Graph Databases

Pablo Barceló    Jorge Pérez    Juan Reutter

Universidad de Chile, PUC Chile
Graph structured data is now everywhere

RDF Linked Data representation of DBLP (real data!)

- DBpedia (RDF representation of Wikipedia)
- Bio2RDF, GeoNames, FreeBase, FOAF, ...
- Facebook, Twitter, ...
Formalisms to exchange graph databases

First define a graph mapping language, then

- Exchanging graph databases
- Computing solutions and answering target queries
- Advanced schema mapping operations
  - composition
  - inversion
  - ...

Outline

Graph mapping language

Computing solutions & answering queries

Composing graph schema mappings
Graph query languages
Graph query languages

\[
\text{RPQ: } \text{partOf} \cdot \text{series}
\]
Graph query languages

\[
\text{RPQ: } \text{partOf} \cdot \text{series}
\]
Graph query languages

2RPQ: creator ← creator
Graph query languages

2RPQ: $\text{creator}^- \cdot \text{creator}$
Graph query languages

2RPQ: $(\text{creator}^- \cdot \text{creator})^*$
Graph query languages

2RPQ: \((\text{creator}^{-} \cdot \text{creator})^*)
Graph query languages

NRE: creator^− · [ partOf · series ] · creator
Graph query languages

NRE: creator⁻¹ · [partOf · series] · creator
Graph query languages

\[
\text{NRE: } \quad (\text{creator} \cdot [\text{partOf} \cdot \text{series}] \cdot \text{creator})^+
\]
Graph query languages

NRE: \((\text{creator}^- \cdot [\text{partOf} \cdot \text{series}] \cdot \text{creator})^+)\)
Conjunctions over RPQs, 2RPQs, and NREs

\[ \exists \bar{y} \left( (u_1, r_1, u'_1) \land \cdots \land (u_k, r_k, u'_k) \right) \]

CRPQs, C2RPQs, CNREs
Graph query languages

\[ \exists u \exists v ( (x, \text{creator}^-, u) \land (u, \text{partOf} \cdot \text{series}, v) \land (u, \text{creator}, y) ) \]
Graph query languages

\[ \exists u \exists v ((x, \text{creator}^-, u) \land (u, \text{partOf} \cdot \text{series}, v) \land (u, \text{creator}, y)) \]
Review on expressiveness

NREs $\not\subseteq$ C2RPQs
(binary) CRPQs $\not\subseteq$ NREs
Review on expressiveness

NREs \nsubseteq C2RPQs
(binary) CRPQs \nsubseteq NREs

Example

\( (\text{creator}^- \cdot [\text{partOf} \cdot \text{series}] \cdot \text{creator})^+ \)

cannot be expressed as a C2RPQ
Review on expressiveness

NREs $\not\subseteq$ C2RPQs
(binary) CRPQs $\not\subseteq$ NREs

Example

$(\text{creator}^- \cdot [\text{partOf} \cdot \text{series}] \cdot \text{creator})^+$
cannot be expressed as a C2RPQ

tree-shaped binary C2RPQs $\equiv$ $(\ )^*-[ ]$ alternation-free NREs
Review on complexity

Evaluation problem for NREs can be solved in $O(|G| \times |\text{expr}|)$ via a PDL-like recursive labeling procedure.

NREs properly extend a linear-time fragment of C2RPQs maintaining the complexity of evaluation.
Review on complexity

Evaluation problem for NREs can be solved in $O(|G| \times |\text{expr}|)$
  - via a PDL-like recursive labeling procedure

NREs properly extends a linear-time fragment of C2RPQs maintaining the complexity of evaluation

Evaluation problem for CRPQs is NP-complete
  - it is in NP for CNREs
Consider two (disjoint) graph alphabets $\Sigma_S$ and $\Sigma_T$

- **Graph mapping:** $\mathcal{M} = (\Sigma_S, \Sigma_T, \mathcal{T})$ s.t. $\mathcal{T}$ contains rules

  $$\varphi_S(\bar{x}) \rightarrow \psi_T(\bar{x})$$

$\varphi_S$ and $\psi_T$ are CNREs over $\Sigma_S$ and $\Sigma_T$, resp.
Consider two (disjoint) graph alphabets $\Sigma_S$ and $\Sigma_T$

- **Graph mapping:** $M = (\Sigma_S, \Sigma_T, \mathcal{T})$ s.t. $\mathcal{T}$ contains rules

  $$\varphi_S(\bar{x}) \rightarrow \psi_T(\bar{x})$$

  $\varphi_S$ and $\psi_T$ are CNREs over $\Sigma_S$ and $\Sigma_T$, resp.

- **$L_1$-to-$L_2$ mapping:** $\varphi_S \in L_1$ and $\psi_T \in L_2$
Graph mapping language

Consider two (disjoint) graph alphabets \( \Sigma_S \) and \( \Sigma_T \)

- **Graph mapping**: \( \mathcal{M} = (\Sigma_S, \Sigma_T, \mathcal{T}) \) s.t. \( \mathcal{T} \) contains rules
  \[
  \varphi_S(\bar{x}) \rightarrow \psi_T(\bar{x})
  \]

  \( \varphi_S \) and \( \psi_T \) are \textit{CNRE}s over \( \Sigma_S \) and \( \Sigma_T \), resp.

- **L_1-to-L_2 mapping**: \( \varphi_S \in L_1 \) and \( \psi_T \in L_2 \)

- **L-GAV mapping**: \( \varphi_S \in L \) and \( \psi_T \) is \((x, a, y)\) with \( a \in \Sigma_T \)
Graph mapping language: example

2RPQ-GAV:

$$(x, (\text{creator}^- \cdot \text{creator})^+, y) \rightarrow (x, \text{connected}, y)$$
Graph mapping language: example

2RPQ-GAV:

\[(\text{creator}^{-} \cdot \text{creator})^+ \rightarrow \text{connected}\]
Graph mapping language: example

2RPQ-GAV:

\[(\text{creator}^{-} \cdot \text{creator})^{+} \rightarrow \text{connected}\]

C2RPQ-to-CRPQ:

\[(y, \text{creator}^{-}, x) \land (x, \text{partOf} \cdot \text{series}, w) \rightarrow (y, \text{makes}, x) \land (x, \text{inConf}, w)\]
Graph mapping language: example

2RPQ-GAV:

$$(\text{creator}^- \cdot \text{creator})^+ \rightarrow \text{connected}$$

C2RPQ-to-CRPQ:

$$(y, \text{creator}^-, x) \land (x, \text{partOf} \cdot \text{series}, w) \rightarrow (y, \text{makes}, x) \land (x, \text{inConf}, w)$$

NRE-GAV:

$$(x, (\text{creator}^- \cdot [\text{partOf} \cdot \text{series}] \cdot \text{creator})^+, y) \rightarrow (x, \text{confConn}, y)$$
Graph mapping language: example

**2RPQ-GAV:**

\[(\text{creator}^- \cdot \text{creator})^+ \rightarrow \text{connected}\]

**C2RPQ-to-CRPQ:**

\[(y, \text{creator}^-, x) \land (x, \text{partOf} \cdot \text{series}, w) \rightarrow (y, \text{makes}, x) \land (x, \text{inConf}, w)\]

**NRE-GAV:**

\[(\text{creator}^\cdot [\text{partOf} \cdot \text{series}] \cdot \text{creator})^+ \rightarrow \text{confConn}\]
Solutions in graph data exchange

- Let $\mathcal{M} = (\Sigma_S, \Sigma_T, \mathcal{T})$ be a graph mapping
- Let $G_S$ be a source graph database
- $G_T$ is a solution for $G_S$ under $\mathcal{M}$ if
  - for every $\varphi_S(\bar{x}) \rightarrow \psi_T(\bar{x})$ in $\mathcal{T}$ and
  - for every tuple $\bar{a}$ of values in $G_S$, we have that

if $\bar{a}$ is in the evaluation of $\varphi_S$ over $G_S$, then $\bar{a}$ is in the evaluation of $\psi_T$ over $G_T$. 
Solutions in graph data exchange

- Let $\mathcal{M} = (\Sigma_\mathcal{S}, \Sigma_\mathcal{T}, \mathcal{T})$ be a graph mapping
- Let $G_\mathcal{S}$ be a source graph database
- $G_\mathcal{T}$ is a solution for $G_\mathcal{S}$ under $\mathcal{M}$ if
  - for every $\varphi_\mathcal{S}(\bar{x}) \rightarrow \psi_\mathcal{T}(\bar{x})$ in $\mathcal{T}$ and
  - for every tuple $\bar{a}$ of values in $G_\mathcal{S}$, we have that
  
  if $\bar{a}$ is in the evaluation of $\varphi_\mathcal{S}$ over $G_\mathcal{S}$, then $\bar{a}$ is in the evaluation of $\psi_\mathcal{T}$ over $G_\mathcal{T}$.

$\text{Sol}_\mathcal{M}(G_\mathcal{S})$ is the set of solutions for $G_\mathcal{S}$ under $\mathcal{M}$. 
Example

\[(y, \text{creator}^-, x) \land (x, \text{partOf} \cdot \text{series}, w) \rightarrow (y, \text{makes}, x) \land (x, \text{inConf}, w)\]
Example

\[(y, \text{creator}^-, x) \land (x, \text{partOf} \cdot \text{series}, w) \implies (y, \text{makes}, x) \land (x, \text{inConf}, w)\]
Example

\[ (\text{makes} \cdot \text{makes}^-)^+ \rightarrow \text{confConnected} \]
Example

\[(\text{makes} \cdot \text{makes}^{-1})^+ \rightarrow \text{confConnected}\]
Interesting expressive power

Example
Copy from source to target all paths of the form

\[ a(aa)^* b \]

classifying the first \( a \) by \( a' \), remaining \( aa \) by \( a'' \), and \( b \) by \( b' \)

We can express this by NRE-mappings
Interesting expressive power

Example

Copy from source to target all paths of the form

\[ a(aa)^* b \]

changing the first \( a \) by \( a' \), remaining \( aa \) by \( aa'' \), and \( b \) by \( b' \)

We can express this by NRE-mappings

\[ a \cdot [(aa)^* b] \rightarrow a' \]
Interesting expressive power

Example

Copy from source to target all paths of the form

$$a(aa)^*b$$

changing the first $$a$$ by $$a'$$, remaining $$aa$$ by $$a''$$, and $$b$$ by $$b'$$

We can express this by NRE-mappings

$$a \cdot [(aa)^* b] \rightarrow a'$$

$$[(a^-a^-)^*a^-] \cdot aa \cdot [(aa)^* b] \rightarrow a''$$
Interesting expressive power

Example

Copy from source to target all paths of the form

\[ a(aa)^* b \]

changing the first \( a \) by \( a' \), remaining \( aa \) by \( a'' \), and \( b \) by \( b' \)

We can express this by NRE-mappings

\[
\begin{align*}
    a \cdot [(aa)^* b] & \rightarrow a' \\
    [(a^- a^-)^* a^-] \cdot aa \cdot [(aa)^* b] & \rightarrow a'' \\
    [(a^- a^-)^* a^-] \cdot b & \rightarrow b'
\end{align*}
\]
Interesting expressive power

Example

Copy from source to target all paths of the form

\[ a(aa)^* b \]

changing the first \( a \) by \( a' \), remaining \( aa \) by \( a'' \), and \( b \) by \( b' \)

We can express this by NRE-mappings

\[ a \cdot [(aa)^* b] \rightarrow a' \]
\[ [(a^- a^-)^* a^-] \cdot aa \cdot [(aa)^* b] \rightarrow a'' \]
\[ [(a^- a^-)^* a^-] \cdot b \rightarrow b' \]

Any regular source path can be \textit{synchronized} in the same way
Outline

Graph mapping language

Computing solutions & answering queries

Composing graph schema mappings
Graph patterns are graphs such that

- Nodes can be labeled with *null values*
- Edges can be labeled with (nested) regular expressions
Graph patterns as universal representatives

Graph patterns are graphs such that

- Nodes can be labeled with *null values*
- Edges can be labeled with (nested) regular expressions

$$\pi: X \xrightarrow{a[a]b^*} n \xleftarrow{b^+} m \xrightarrow{b}$$
Graph patterns: semantics

Semantics of graph patterns in terms of homomorphisms:

Given a pattern $\pi$, graph database $G$ is in $\text{rep}(\pi)$ iff there exists homomorphism $h$ from nulls in $\pi$ to nodes in $G$ s.t.

for every $(u, expr, v)$ in $\pi$ there is a path in $G$ from $h(u)$ to $h(v)$ that satisfies $expr$. 
Graph patterns: semantics

Semantics of graph patterns in terms of homomorphisms:

Given a pattern $\pi$, graph database $G$ is in $\text{rep}(\pi)$ iff there exists homomorphism $h$ from nulls in $\pi$ to nodes in $G$ s.t.

for every $(u, \text{expr}, v)$ in $\pi$ there is a path in $G$ from $h(u)$ to $h(v)$ that satisfies $\text{expr}$.

\[ \begin{array}{c}
\pi: \\
X & \xrightarrow{a[a]b^*} & n \\
& \xrightarrow{b^+} & m \\
& \xrightarrow{b} & \\
G: \\
\end{array} \]
Computing universal representatives

**Definition**

\( \pi_T \) is a *universal representative* for graph \( G_S \) under \( \mathcal{M} \) if

\[
\text{Sol}_{\mathcal{M}}(G_S) = \text{rep}(\pi_T)
\]
Computing universal representatives

**Definition**

πₜ is a universal representative for graph Gₛ under M if

\[ \text{Sol}_M(Gₛ) = \text{rep}(\piₜ) \]

**Proposition**

- Given graph Gₛ and mapping M, a universal representative always exists and can be computed in polynomial space
- For fixed M it can be computed in polynomial time

just a simple adaptation of the chase procedure...
Feasible universal representative computation

Universal representatives can be in general of size exponential in the size of the mapping

Proposition

*Computing universal representatives is $\text{FP}^{\text{NP}[\log]}$-hard even restricted to inputs ensuring univ representatives of polynomial size*
Feasible universal representative computation

Universal representatives can be in general of size exponential in the size of the mapping

Proposition

*Computing universal representatives is \( \text{FP}^{\text{NP[log]}} \)-hard even restricted to inputs ensuring univ representatives of polynomial size*

Proposition

*Given NRE-to-CNRE mapping \( \mathcal{M} \) a universal representative can be computed in \( O(|G_s|^2 \times |\mathcal{M}|) \) (tight bound)*
Certain answers

Definition

$$\text{certain}_M(Q_T, G_S) = \bigcap_{G_T \in \text{Sol}_M(G_S)} Q_T(G_T)$$
Certain answers

Definition

\[
certain_{\mathcal{M}}(Q_T, G_S) = \bigcap_{G_T \in \text{Sol}_{\mathcal{M}}(G_S)} Q_T(G_T)
\]

Observation: if \( \pi_T \) is a universal representative, then

\[
certain_{\mathcal{M}}(Q_T, G_S) = \bigcap_{G_T \in \text{rep}(\pi_T)} Q_T(G_T)
\]
Certain answers

**Definition**

\[
\text{certain}_M(Q_T, G_S) = \bigcap_{G_T \in \text{Sol}_M(G_S)} Q_T(G_T)
\]

**Observation:** if \( \pi_T \) is a universal representative, then

\[
\text{certain}_M(Q_T, G_S) = \bigcap_{G_T \in \text{rep}(\pi_T)} Q_T(G_T)
\]

**CertAns**

Input: Graph \( G_S \), mapping \( M \), target query \( Q_T \), and tuple \( \bar{a} \)

Output: Is \( \bar{a} \) in \( \text{certain}_M(Q_T, G_S) \)?
Complexity of computing certain answers

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) <strong>CertAns</strong> is in EXPSPACE for CNRE-to-CNRE mappings and CNRE queries</td>
</tr>
<tr>
<td>(2) <strong>CertAns</strong> is EXPSPACE-hard for CRPQ-to-CRPQ mappings and CRPQ queries</td>
</tr>
</tbody>
</table>
Complexity of computing certain answers

Theorem

1. **CertAns** is in EXPSPACE for CNRE-to-CNRE mappings and CNRE queries
2. **CertAns** is EXPSPACE-hard for CRPQ-to-CRPQ mappings and CRPQ queries

- (2) follows from known EXPSPACE-hard complexity of query containment for CRPQs (Calvanese et al.)
Complexity of computing certain answers

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) <strong>CertAns</strong> is in EXPSPACE for CNRE-to-CNRE mappings and CNRE queries</td>
</tr>
<tr>
<td>(2) <strong>CertAns</strong> is EXPSPACE-hard for CRPQ-to-CRPQ mappings and CRPQ queries</td>
</tr>
</tbody>
</table>

- (2) follows from known EXPSPACE-hard complexity of query containment for CRPQs (Calvanese et al.)
- (1) needed the adaptation of techniques in (Calvanese et al.):
Complexity of computing certain answers

**Theorem**

1. \textbf{CertAns} is in EXPSPACE for CNRE-to-CNRE mappings and CNRE queries

2. \textbf{CertAns} is EXPSPACE-hard for CRPQ-to-CRPQ mappings and CRPQ queries

- (2) follows from known EXPSPACE-hard complexity of query containment for CRPQs (Calvanese et al.)
- (1) needed the adaptation of techniques in (Calvanese et al.):
  
  \textit{Alternating 2-way automata to represent canonical solutions}
Complexity of computing certain answers

**Theorem**

(1) \text{CertAns} \textit{is in EXPSPACE} for CNRE-to-CNRE mappings and CNRE queries

(2) \text{CertAns} \textit{is EXPSPACE-hard} for CRPQ-to-CRPQ mappings and CRPQ queries

- (2) follows from known EXPSPACE-hard complexity of query containment for CRPQs (Calvanese et al.)
- (1) needed the adaptation of techniques in (Calvanese et al.):
  
  \textit{Alternating 2-way automata to represent canonical solutions}

\[ e_1 \cdot [e_2] \cdot [e_3] \cdot (e_4 \cdot [e_5])^* \]
Complexity of computing certain answers

Theorem

(1) CertAns is in EXPSPACE for CNRE-to-CNRE mappings and CNRE queries

(2) CertAns is EXPSPACE-hard for CRPQ-to-CRPQ mappings and CRPQ queries

◮ (2) follows from known EXPSPACE-hard complexity of query containment for CRPQs (Calvanese et al.)
◮ (1) needed the adaptation of techniques in (Calvanese et al.):

Alternating 2-way automata to represent canonical solutions

\[ e_1 \cdot [e_2] \cdot [e_3] \cdot (e_4 \cdot [e_5])^* \]
Complexity of computing certain answers

**Theorem**

1. \text{CertAns} is in \text{EXPSPACE} for CNRE-to-CNRE mappings and CNRE queries
2. \text{CertAns} is \text{EXPSPACE-hard} for CRPQ-to-CRPQ mappings and CRPQ queries

- (2) follows from known \text{EXPSPACE-hard} complexity of query containment for CRPQs (Calvanese et al.)
- (1) needed the adaptation of techniques in (Calvanese et al.): *Alternating 2-way automata to represent canonical solutions*

\[
e_1 \cdot [e_2] \cdot [e_3] \cdot (e_4 \cdot [e_5])^* \]

need to run over (a restricted class of) trees
Even data complexity is hard

\[ \text{CERTAns}(\mathcal{M}, Q_T) \]

**Input:** Graph $G_S$, and tuple $\bar{a}$

**Output:** Is $\bar{a}$ in $\text{certain}_\mathcal{M}(Q_T, G_S)$?
Even data complexity is hard

**CertAns**($\mathcal{M}, Q_T$)

- **Input:** Graph $G_S$, and tuple $\bar{a}$
- **Output:** Is $\bar{a}$ in $\text{Cert}_{\mathcal{M}}(Q_T, G_S)$?

**Theorem**

1. **CertAns**($\mathcal{M}, Q_T$) is coNP-complete for every CNRE-to-CNRE mapping and CNRE query.
2. **CertAns**($\mathcal{M}, Q_T$) is coNP-hard even for RPQ-to-RPQ mappings and RPQ queries.
Even data complexity is hard

**CertAns(\(M, QT\))**

- **Input:** Graph \(G_S\), and tuple \(\vec{a}\)
- **Output:** Is \(\vec{a}\) in certain \(M(Q_T, G_S)\)?

**Theorem**

1. **CertAns(\(M, QT\))** is coNP-complete for every CNRE-to-CNRE mapping and CNRE query.
2. **CertAns(\(M, QT\))** is coNP-hard even for RPQ-to-RPQ mappings and RPQ queries.

In the paper:
- Structural properties ensuring tractable data complexity
Tractable query answering

High complexity if we allow conjunctions in rules or regular expressions in the right-side

▶ Need to focus on GAV mappings.
Tractable query answering

High complexity if we allow conjunctions in rules or regular expressions in the right-side

- Need to focus on GAV mappings.

By just computing a universal representative we obtain

Corollary

For NRE-GAV mappings and NRE queries, \textsc{CertAns} can be solved in time

\[ O(|G_s|^2 \times |\mathcal{M}| \times |\text{expr}|) \]
Tractable query answering

High complexity if we allow conjunctions in rules or regular expressions in the right-side

▶ Need to focus on GAV mappings.

By just computing a universal representative we obtain

**Corollary**

*For NRE-GAV mappings and NRE queries, $\text{CERTANS}$ can be solved in time*

$$O(|G_s|^2 \times |\mathcal{M}| \times |\text{expr}|)$$

But we can do better
Tractable query answering

High complexity if we allow conjunctions in rules or regular expressions in the right-side

- Need to focus on GAV mappings.

By just computing a universal representative we obtain

**Corollary**

For NRE-GAV mappings and NRE queries, $\text{CertAns}$ can be solved in time

$$O(|G_s|^2 \times |M| \times |expr|)$$

But we can do better

**Theorem**

For NRE-GAV mappings and NRE queries, $\text{CertAns}$ can be solved in time

$$O(|G_s| \times |M| \times |expr|)$$
Outline

Graph mapping language

Computing solutions & answering queries

Composing graph schema mappings
Composing mappings

\[
S_A \quad S_B \quad S_C
\]
Composing mappings

$S_A \xrightarrow{\mathcal{M}_{AB}} S_B \xrightarrow{\mathcal{M}_{BC}} S_C$
Composing mappings

\[ M_{AB} \rightarrow M_{BC} \rightarrow M_{AC} \]
Composing mappings

Intuitively, $M_{AC}$ must have the same effect as applying $M_{AB}$ and then $M_{BC}$

$$M_{AC} = M_{AB} \circ M_{BC}$$
Composing mappings

\[ \mathcal{M}_{AC} = \mathcal{M}_{AB} \circ \mathcal{M}_{BC} \]

Intuitively, \( \mathcal{M}_{AC} \) must have the same effect as applying \( \mathcal{M}_{AB} \) and then \( \mathcal{M}_{BC} \)

- how to compute the composition?
- what is the language needed to express it?
- is there a language closed under composition?
CRPQs are not suitable for composing graph mappings
CRPQs are not suitable for composing graph mappings

Example

\[ M_1: \exists u \ (x, \text{creator}^-, y) \land (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y) \]
CRPQs are not suitable for composing graph mappings

Example

$M_1: \exists u \ (x, \text{creator}^-, y) \land (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y)$

$M_2: (x, (\text{confAuthor} \cdot \text{confAuthor}^-)^+, y) \rightarrow (x, \text{confConnected}, y)$
CRPQs are not suitable for composing graph mappings

Example

\[ M_1: \exists u \ (x, \text{creator}^-, y) \land (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y) \]

\[ M_2: (x, (\text{confAuthor} \cdot \text{confAuthor}^-)^+, y) \rightarrow (x, \text{confConnected}, y) \]

\[ M_1 \circ M_2 \]
CRPQs are not suitable for composing graph mappings

Example

$$M_1: \exists u \ (x, \text{creator}^-, y) \wedge (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y)$$

$$M_2: (x, (\text{confAuthor} \cdot \text{confAuthor}^-)^+, y) \rightarrow (x, \text{confConnected}, y)$$

$$M_1 \circ M_2$$

Example

$$M_1: (x, \text{creator}^- \cdot [\ \text{partOf} \cdot \text{series }], y) \rightarrow (x, \text{confAuthor}, y)$$
CRPQs are not suitable for composing graph mappings

Example

\[ M_1: \exists u \ (x, \text{creator}^{-}, y) \land (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y) \]

\[ M_2: (x, (\text{confAuthor} \cdot \text{confAuthor}^{-})^{+}, y) \rightarrow (x, \text{confConnected}, y) \]

\[ M_1 \circ M_2 \]

Example

\[ M_1: (x, \text{creator}^{-} \cdot [\text{partOf} \cdot \text{series}], y) \rightarrow (x, \text{confAuthor}, y) \]

\[ M_2: (x, (\text{confAuthor} \cdot \text{confAuthor}^{-})^{+}, y) \rightarrow (x, \text{confConnected}, y) \]
CRPQs are not suitable for composing graph mappings

Example

\[ M_1: \exists u \ (x, \text{creator}^-, y) \land (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y) \]

\[ M_2: (x, (\text{confAuthor} \cdot \text{confAuthor}^-)^+, y) \rightarrow (x, \text{confConnected}, y) \]

\[ M_1 \circ M_2 ??? \]

Example

\[ M_1: \text{creator}^- \cdot [\text{partOf} \cdot \text{series}] \rightarrow \text{confAuthor} \]

\[ M_2: (\text{confAuthor} \cdot \text{confAuthor}^-)^+ \rightarrow \text{confConnected} \]
CRPQs are not suitable for composing graph mappings

**Example**

\[ M_1: \exists u \ (x, creator^-, y) \land (y, partOf \cdot series, u) \rightarrow (x, confAuthor, y) \]

\[ M_2: \ (x, (confAuthor \cdot confAuthor^-)^+, y) \rightarrow (x, confConnected, y) \]

\[ M_1 \circ M_2 \]???

**Example**

\[ M_1: \ creator^- \cdot [partOf \cdot series] \rightarrow confAuthor \]

\[ M_2: \ (confAuthor \cdot confAuthor^-)^+ \rightarrow confConnected \]

\[ M_1 \circ M_2: \ ((creator^- \cdot [partOf \cdot series]) \cdot ([partOf \cdot series] \cdot creator))^+ \rightarrow confConnected \]
CRPQs are not suitable for composing graph mappings

Example

\[ M_1: \exists u \ (x, \text{creator}^-, y) \land (y, \text{partOf} \cdot \text{series}, u) \rightarrow (x, \text{confAuthor}, y) \]

\[ M_2: \ (x, (\text{confAuthor} \cdot \text{confAuthor}^-)^+, y) \rightarrow (x, \text{confConnected}, y) \]

\[ M_1 \circ M_2 \]

Example

\[ M_1: \ 	ext{creator}^- \cdot [\ \text{partOf} \cdot \text{series} ] \rightarrow \text{confAuthor} \]

\[ M_2: \ (\text{confAuthor} \cdot \text{confAuthor}^-)^+ \rightarrow \text{confConnected} \]

\[ M_1 \circ M_2: \ (\text{creator}^- \cdot [\ \text{partOf} \cdot \text{series} ] \cdot \text{creator})^+ \rightarrow \text{confConnected} \]
NRE-GAV mappings are closed under composition

**Theorem**

*The composition of NRE-GAV mappings can always be specified by an NRE-GAV mapping*
NRE-GAV mappings are closed under composition

**Theorem**

*The composition of NRE-GAV mappings can always be specified by an NRE-GAV mapping*

**Corollary**

*The composition of tree-shaped C2RPQ-GAV mappings can always be specified by an NRE-GAV mapping*
Known result in relational data exchange:

- CQ-GAV mappings are closed under composition
Composition in the presence of conjunctions

Known result in relational data exchange:

- CQ-GAV mappings are closed under composition

Proposition

There exist CRPQ-GAV mappings s.t. their composition cannot be specified by a CNRE-GAV mapping
Composition in the presence of conjunctions

Known result in relational data exchange:
- CQ-GAV mappings are closed under composition

**Proposition**

*There exist CRPQ-GAV mappings s.t. their composition cannot be specified by a CNRE-GAV mapping*

Open question:
What is the language needed to compose CRPQ-GAV mappings?
Concluding remarks

We have initiated the study of Graph Data Exchange

- Some techniques can be adapted from the relational case
- Query answering is highly complex
- Schema mapping operators is a challenging topic
- NREs add expressive power compared with 2RPQs maintaining the complexity plus giving good properties for composition
Concluding remarks

We have initiated the study of Graph Data Exchange

- Some techniques can be adapted from the relational case
- Query answering is highly complex
- Schema mapping operators is a challenging topic
- NREs add expressive power compared with 2RPQs maintaining the complexity plus giving good properties for composition

We would like to explore new formalisms to specify mappings

- Can we add expressive power maintaining the complexity?
Concluding remarks

We have initiated the study of Graph Data Exchange

- Some techniques can be adapted from the relational case
- Query answering is highly complex
- Schema mapping operators is a challenging topic
- NREs add expressive power compared with 2RPQs maintaining the complexity plus giving good properties for composition

We would like to explore new formalisms to specify mappings

- Can we add expressive power maintaining the complexity?
  - Good candidate to start: GraphXPath
Concluding remarks

We have initiated the study of Graph Data Exchange

- Some techniques can be adapted from the relational case
- Query answering is highly complex
- Schema mapping operators is a challenging topic
- NREs add expressive power compared with 2RPQs maintaining the complexity plus giving good properties for composition

We would like to explore new formalisms to specify mappings

- Can we add expressive power maintaining the complexity?
  - Good candidate to start: GraphXPath
- More natural (and expressive) synchronization between paths

\[(a/a')(aa/a'')^*(b/b')\]
Outline

Graph mapping language

Computing solutions & answering queries

Composing graph schema mappings