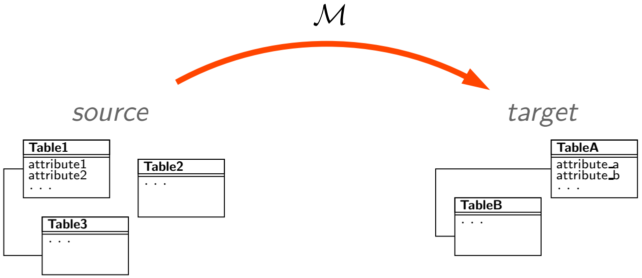
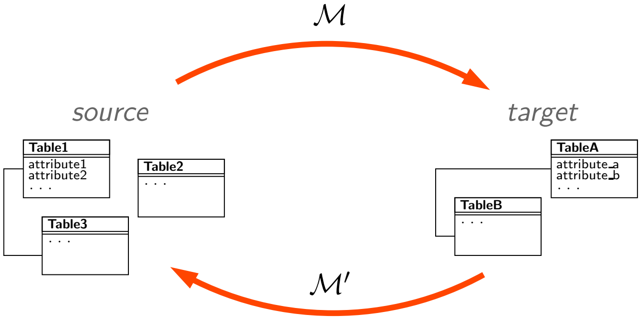
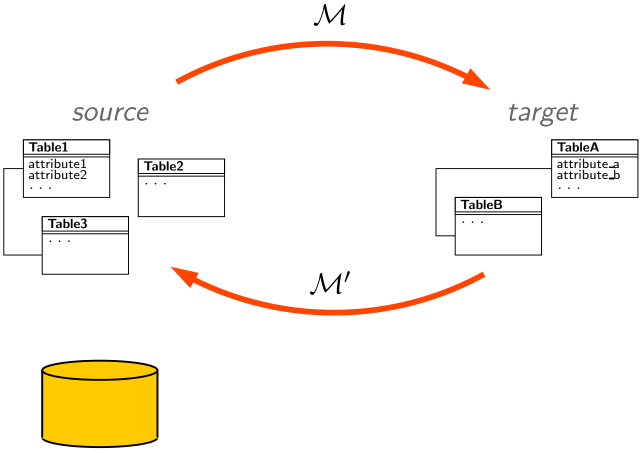
Invertion is a fundamental operation for schema mapping management.



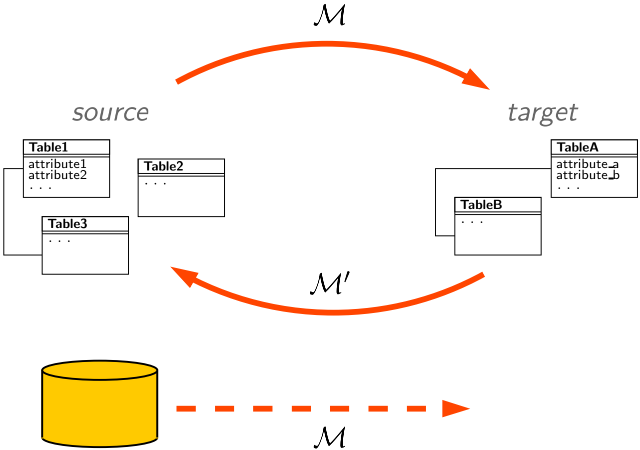
Invertion is a fundamental operation for schema mapping management.



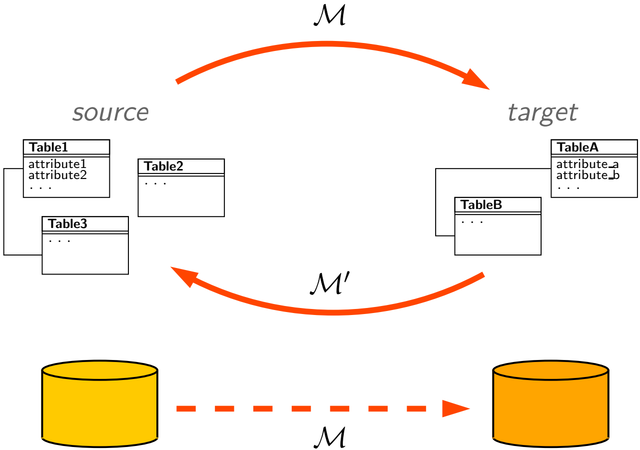
Invertion is a fundamental operation for schema mapping management.



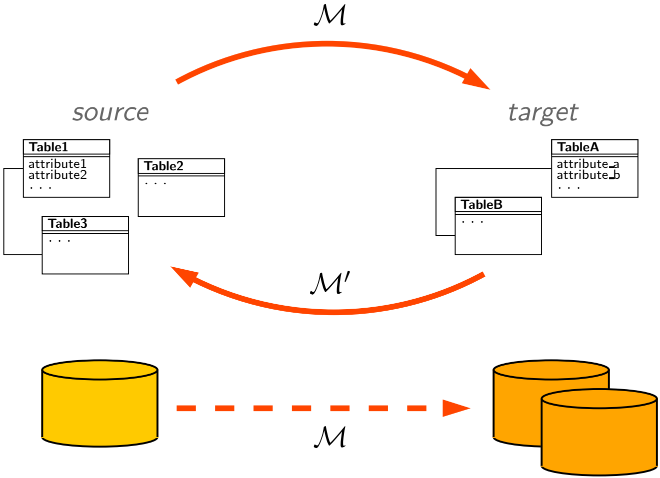
Invertion is a fundamental operation for schema mapping management.



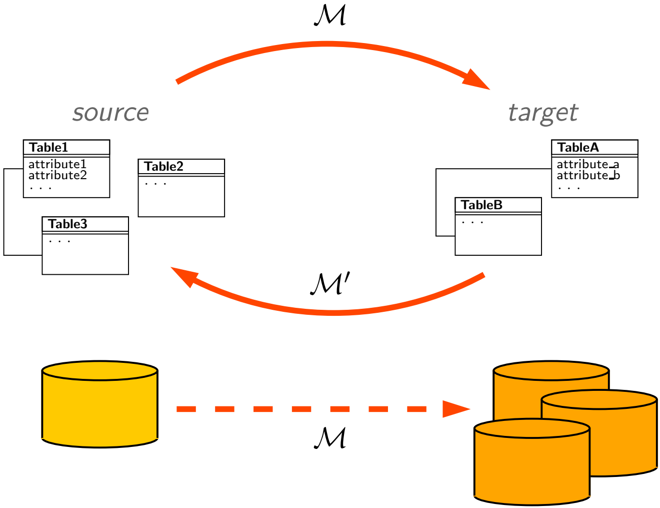
Invertion is a fundamental operation for schema mapping management.



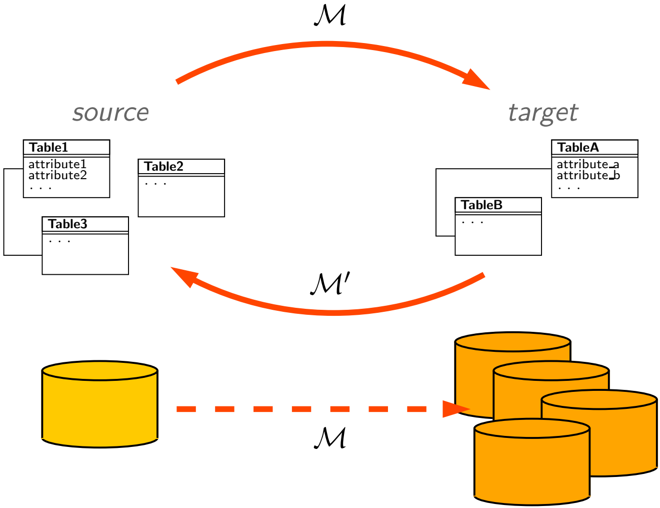
Invertion is a fundamental operation for schema mapping management.



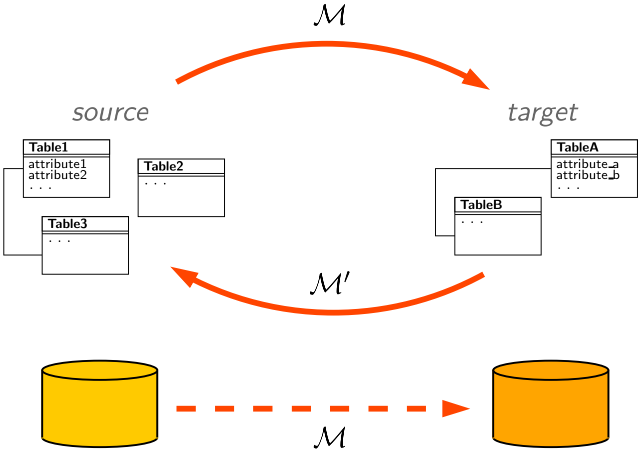
Invertion is a fundamental operation for schema mapping management.



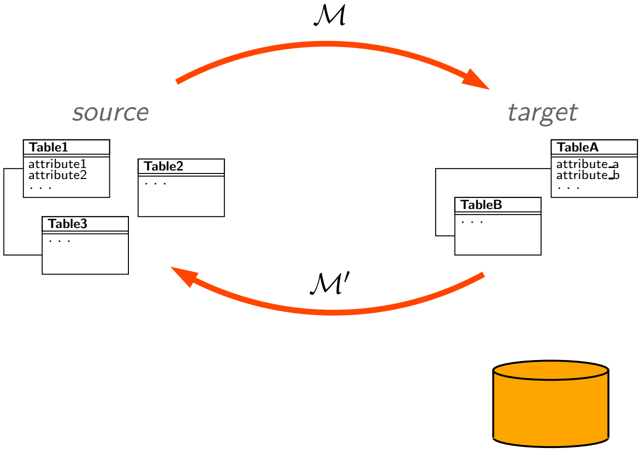
Invertion is a fundamental operation for schema mapping management.



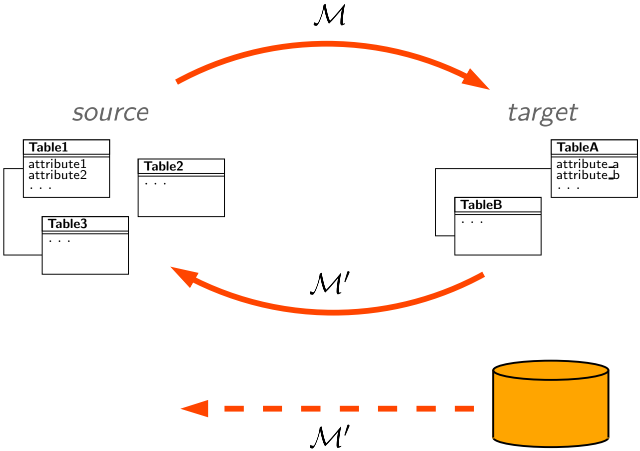
Invertion is a fundamental operation for schema mapping management.



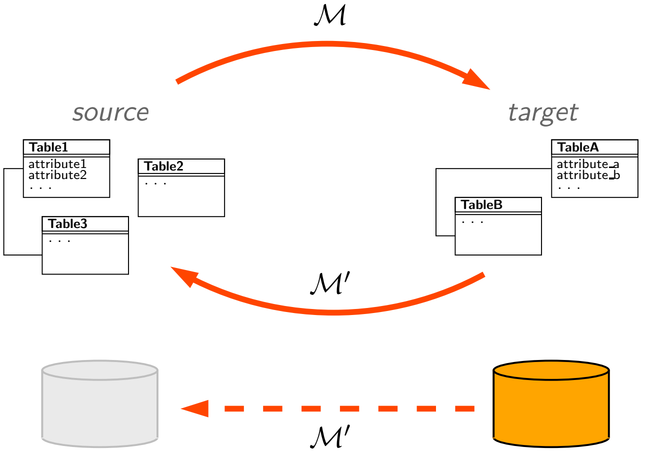
Invertion is a fundamental operation for schema mapping management.



Invertion is a fundamental operation for schema mapping management.



Invertion is a fundamental operation for schema mapping management.



The study of the inverse operator has mainly focused on foundational issues.

Fagin-Inverse, Quasi-Inverse, Maximum Recovery:

- Have focused on the (important) problem of defining a good semantics for inversion.

The study of the inverse operator has mainly focused on foundational issues.

Fagin-Inverse, Quasi-Inverse, Maximum Recovery:

- Have focused on the (important) problem of defining a good semantics for inversion.

From a practical point of view they have some problems:

- Some are too restrictive.

The study of the inverse operator has mainly focused on foundational issues.

Fagin-Inverse, Quasi-Inverse, Maximum Recovery:

- Have focused on the (important) problem of defining a good semantics for inversion.

From a practical point of view they have some problems:

- Some are too restrictive.
- Some need too much expressive power to be specified (far from practical settings).

The study of the inverse operator has mainly focused on foundational issues.

Fagin-Inverse, Quasi-Inverse, Maximum Recovery:

- Have focused on the (important) problem of defining a good semantics for inversion.

From a practical point of view they have some problems:

- Some are too restrictive.
- Some need too much expressive power to be specified (far from practical settings).
- All the algorithms proposed so far work in exponential time.

The study of the inverse operator has mainly focused on foundational issues.

Fagin-Inverse, Quasi-Inverse, Maximum Recovery:

- Have focused on the (important) problem of defining a good semantics for inversion.

From a practical point of view they have some problems:

- Some are too restrictive.
- Some need too much expressive power to be specified (far from practical settings).
- All the algorithms proposed so far work in exponential time.

We propose solutions to the above issues.

Inverting Schema Mappings: Bridging the Gap between Theory and Practice

Marcelo Arenas, Jorge Pérez, Cristian Riveros, Juan Reutter

Computer Science Department, PUC – Chile

Main results

1. A new notion of inverse based on queries
2. A proof that when focusing on **CQ**, inverses of tgds can be expressed in a language that has the same good properties as tgds for data exchange (we provide an algorithm).
3. Algorithm for computing all the previous notions of inverse in polynomial time (drawback: uses an SO language)

Main results

1. A new notion of inverse based on queries
2. A proof that when focusing on **CQ**, inverses of tgds can be expressed in a language that has the same good properties as tgds for data exchange (we provide an algorithm).
3. Algorithm for computing all the previous notions of inverse in polynomial time (drawback: uses an SO language)

In this talk only 1 and 2

A bit of notation...

A *mapping* $\mathcal{M}$ is a set of pairs (I, J) with

- I a source instance,
- J a target instance.

A bit of notation...

A *mapping* $\mathcal{M}$ is a set of pairs (I, J) with

- I a source instance,
- J a target instance.

The *composition* $\mathcal{M} \circ \mathcal{M}'$ is the set of pairs (I_1, I_2) such that:

- there exists J with $(I_1, J) \in \mathcal{M}$ and $(J, I_2) \in \mathcal{M}'$.

A bit of notation...

A *mapping* $\mathcal{M}$ is a set of pairs (I, J) with

- I a source instance,
- J a target instance.

The *composition* $\mathcal{M} \circ \mathcal{M}'$ is the set of pairs (I_1, I_2) such that:

- there exists J with $(I_1, J) \in \mathcal{M}$ and $(J, I_2) \in \mathcal{M}'$.

If $(I, J) \in \mathcal{M}$ then J *is a solution for* I under $\mathcal{M}$

- $J \in \mathrm{Sol}_{\mathcal{M}}(I)$.

A bit of notation...

A *mapping* $\mathcal{M}$ is a set of pairs (I, J) with

- I a source instance,
- J a target instance.

The *composition* $\mathcal{M} \circ \mathcal{M}'$ is the set of pairs (I_1, I_2) such that:

- there exists J with $(I_1, J) \in \mathcal{M}$ and $(J, I_2) \in \mathcal{M}'$.

If $(I, J) \in \mathcal{M}$ then J *is a solution for* I under $\mathcal{M}$

- $J \in \mathrm{Sol}_{\mathcal{M}}(I)$.

We say that $\mathcal{M}$ is specified by a set of formulas Σ if

- $(I, J) \in \mathcal{M}$ iff (I, J) satisfies Σ.

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathsf{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned} \mathcal{M}: &\quad A(x,y) \rightarrow \exists Z \; T(x,Z) \wedge R(Z,y) \\ I: &\quad \{ \; A(1,1) \; \} \end{aligned}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned} \mathcal{M}: &\quad A(x,y) \rightarrow \exists Z\ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: &\quad \{\ A(1,1)\ \} \end{aligned}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned} \mathcal{M}: &\quad A(x,y) \rightarrow \exists Z\ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: &\quad \{\ A(1,1)\ \} \\ \mathsf{Sol}_{\mathcal{M}}(I): &\end{aligned}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned} \mathcal{M}: &\quad A(x,y) \rightarrow \exists Z\ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: &\quad \{\ A(1,1)\ \} \\ \mathrm{Sol}_{\mathcal{M}}(I): &\quad \{T(1,1), R(1,1)\} \end{aligned}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{array}{rl} \mathcal{M}: & A(x,y) \rightarrow \exists Z\ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: & \{\ A(1,1)\ \} \\ \mathsf{Sol}_{\mathcal{M}}(I): & \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\} \end{array}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned} \mathcal{M}: &\quad A(x,y) \rightarrow \exists Z\ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: &\quad \{\ A(1,1)\ \} \\ \mathrm{Sol}_{\mathcal{M}}(I): &\quad \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots \end{aligned}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned}
\mathcal{M} :&\quad A(x,y) \;\rightarrow\; \exists Z\; T(x,Z) \wedge R(Z,y) \qquad\qquad \leftarrow \text{tgd} \\
I :&\quad \{\; A(1,1) \;\} \\
\mathrm{Sol}_{\mathcal{M}}(I) :&\quad \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\bot), R(\bot,1)\}, \ldots
\end{aligned}$$

$$Q_1(u) : \;\; \exists Z\; T(u,Z) \;\wedge\; \exists V\; R(V,u)$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{array}{rl}
\mathcal{M}: & A(x,y) \rightarrow \exists Z \; T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\
I: & \{ \; A(1,1) \; \} \\
\mathrm{Sol}_{\mathcal{M}}(I): & \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots \\
 & \\
Q_1(u): & \exists Z \; T(u,Z) \; \wedge \; \exists V \; R(V,u) \qquad \leftarrow \textbf{CQ}
\end{array}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{array}{rl} \mathcal{M}: & A(x,y) \rightarrow \exists Z \; T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: & \{ \; A(1,1) \; \} \\ \mathrm{Sol}_{\mathcal{M}}(I): & \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots \end{array}$$

$$Q_1(u): \; \exists Z \; T(u,Z) \wedge \exists V \; R(V,u) \qquad \leftarrow \textbf{CQ}$$

$$\underline{\mathrm{certain}}_{\mathcal{M}}(Q_1, I) = \{1\}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned}
\mathcal{M}: &\quad A(x, y) \ \rightarrow \ \exists Z \ T(x, Z) \wedge R(Z, y) \qquad\qquad \leftarrow \text{tgd} \\
I: &\quad \{\ A(1, 1)\ \} \\
\mathrm{Sol}_{\mathcal{M}}(I): &\quad \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots
\end{aligned}$$

$$Q_2(u): \ \exists Z \ R(Z, u) \wedge Z \neq u$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{array}{rll} \mathcal{M}: & A(x,y) \ \rightarrow \ \exists Z \ T(x,Z) \wedge R(Z,y) & \leftarrow \text{tgd} \\ I: & \{ \ A(1,1) \ \} & \\ \mathrm{Sol}_{\mathcal{M}}(I): & \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\bot), R(\bot,1)\}, \ldots & \\ Q_2(u): & \exists Z \ R(Z,u) \wedge Z \neq u & \leftarrow \mathbf{CQ}^{\neq} \end{array}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned} \mathcal{M} :&\quad A(x,y) \;\rightarrow\; \exists Z \; T(x,Z) \wedge R(Z,y) \qquad\qquad \leftarrow \text{tgd} \\ I :&\quad \{\, A(1,1) \,\} \\ \mathrm{Sol}_{\mathcal{M}}(I) :&\quad \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots \end{aligned}$$

$$Q_2(u) : \;\; \exists Z \; R(Z,u) \wedge Z \neq u \qquad\qquad \leftarrow \mathbf{CQ}^{\neq}$$

$$\underline{\text{certain}}_{\mathcal{M}}(Q_2, I) = \{\}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathsf{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{aligned}
\mathcal{M}: &\quad A(x,y) \ \rightarrow \ \exists Z \ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\
I: &\quad \{ \ A(1,1) \ \} \\
\mathsf{Sol}_{\mathcal{M}}(I): &\quad \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots \\
Q_3(u): &\quad T(u,u) \ \vee \ \exists Z \ R(Z,u) \wedge Z \neq u
\end{aligned}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathsf{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\mathsf{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

$$\begin{array}{rll}
\mathcal{M}: & A(x,y) \rightarrow \exists Z \; T(x,Z) \wedge R(Z,y) & \leftarrow \text{tgd} \\
I: & \{ \; A(1,1) \; \} & \\
\mathsf{Sol}_{\mathcal{M}}(I): & \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots & \\
Q_3(u): & T(u,u) \vee \exists Z \; R(Z,u) \wedge Z \neq u & \leftarrow \mathbf{UCQ}^{\neq}
\end{array}$$

Certain answers: tuples present in all the solutions

Given a mapping $\mathcal{M}$ and a source instance I

$\bar{a}$ is a *certain answer* for Q over I iff

$$\bar{a} \in \bigcap_{J \in \mathrm{Sol}_{\mathcal{M}}(I)} Q(J)$$

We denote by $\underline{\text{certain}}_{\mathcal{M}}(Q, I)$ the set of certain answers

Example

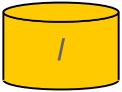
$$\begin{array}{rl} \mathcal{M}: & A(x,y) \rightarrow \exists Z\ T(x,Z) \wedge R(Z,y) \qquad \leftarrow \text{tgd} \\ I: & \{\ A(1,1)\ \} \\ \mathrm{Sol}_{\mathcal{M}}(I): & \{T(1,1), R(1,1)\}, \{T(1,2), R(2,1)\}, \{T(1,\perp), R(\perp,1)\}, \ldots \end{array}$$

$$Q_3(u): \quad T(u,u) \ \vee\ \exists Z\ R(Z,u) \wedge Z \neq u \qquad \leftarrow \mathbf{UCQ}^{\neq}$$

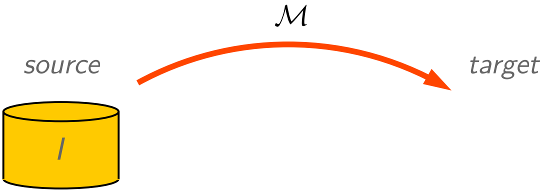
$$\underline{\text{certain}}_{\mathcal{M}}(Q_3, I) = \{1\}$$

Recovering sound information wrt a query

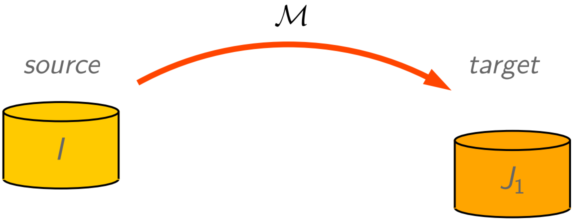
source *target*



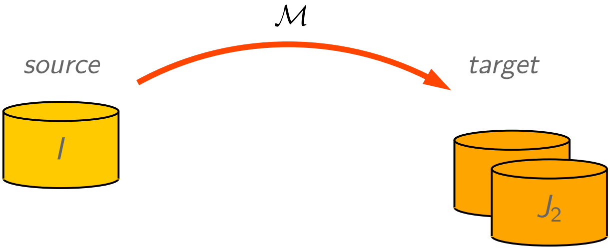
Recovering sound information wrt a query



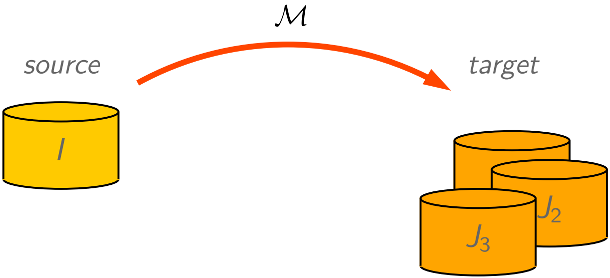
Recovering sound information wrt a query



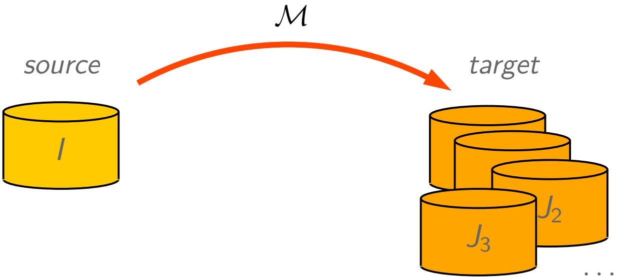
Recovering sound information wrt a query



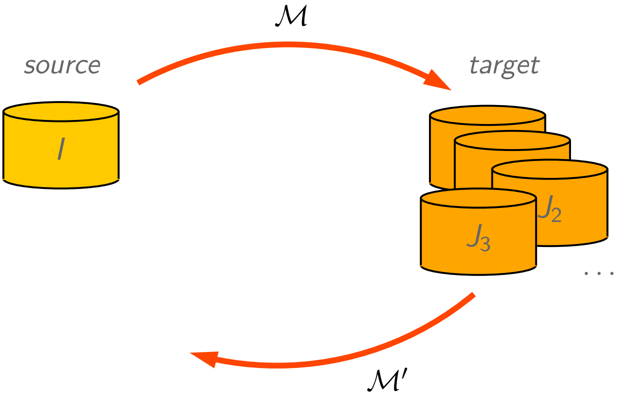
Recovering sound information wrt a query



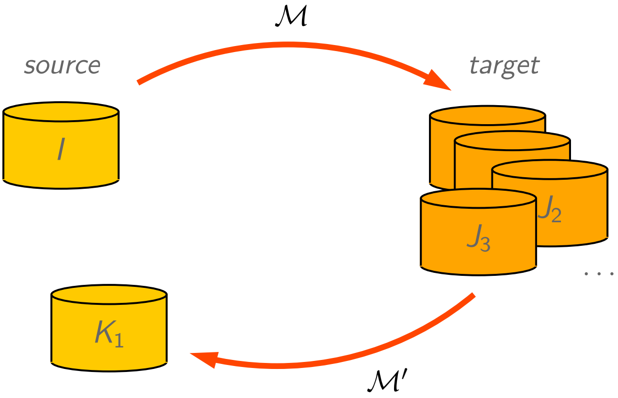
Recovering sound information wrt a query



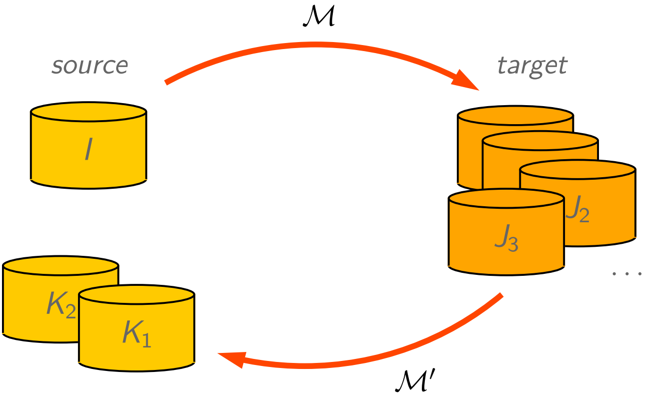
Recovering sound information wrt a query



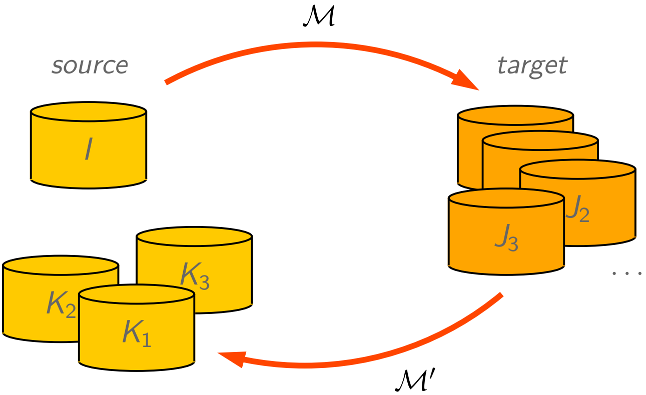
Recovering sound information wrt a query



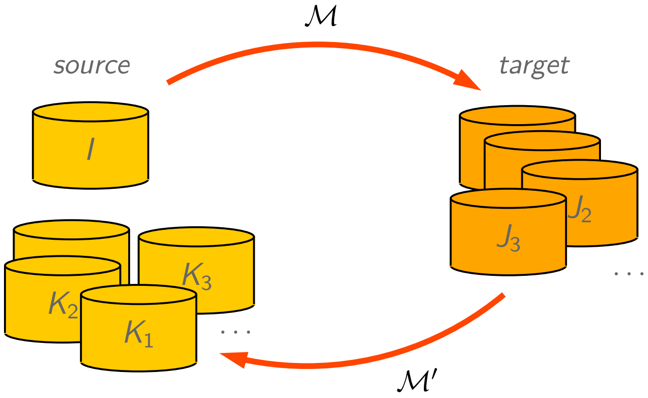
Recovering sound information wrt a query



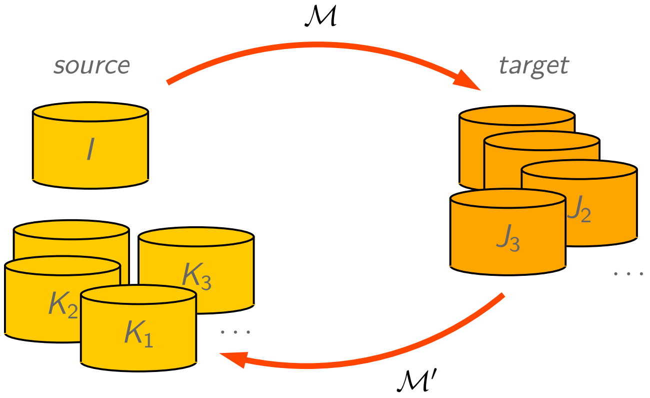
Recovering sound information wrt a query



Recovering sound information wrt a query

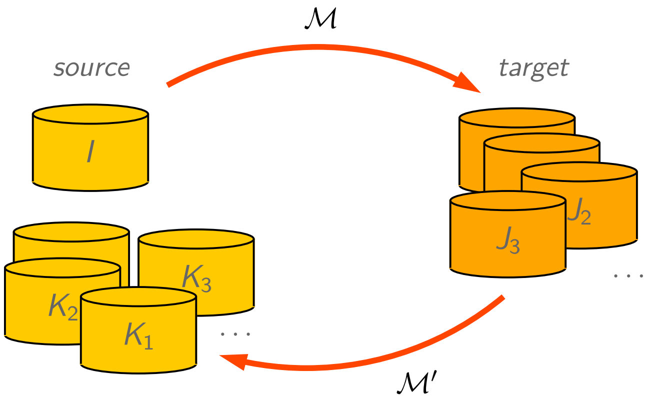


Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

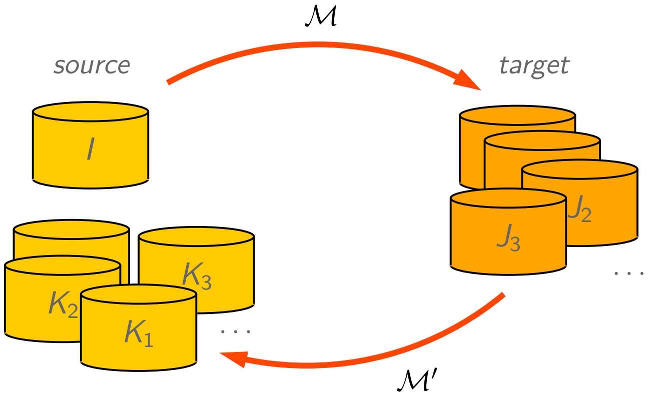
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1)$$

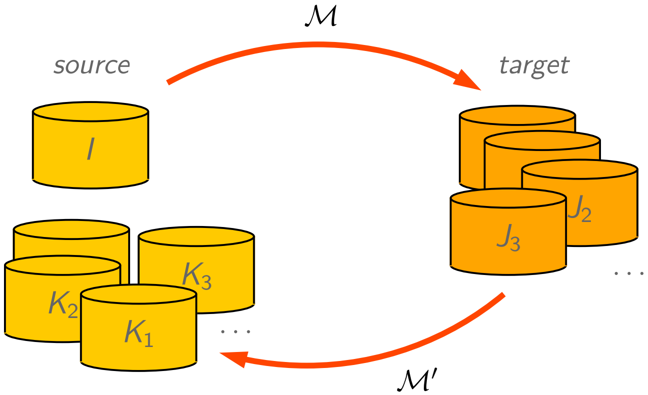
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap$$

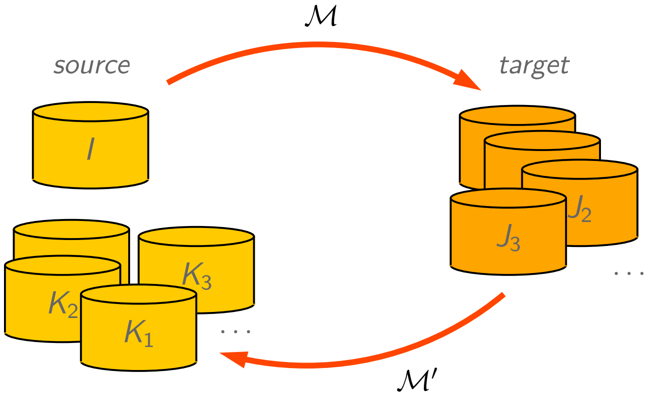
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap Q(K_2)$$

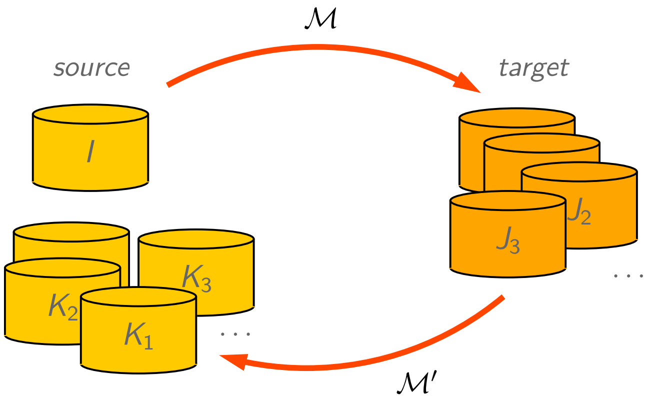
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap Q(K_2) \cap$$

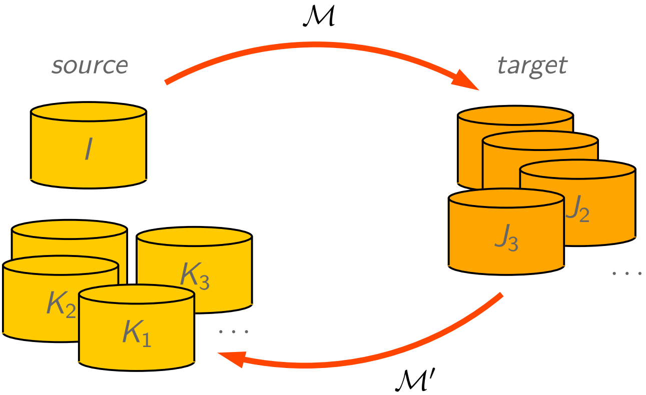
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap Q(K_2) \cap Q(K_3)$$

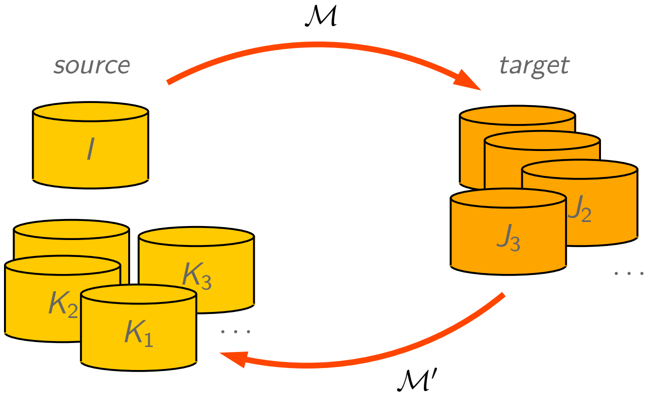
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap Q(K_2) \cap Q(K_3) \cap \cdots$$

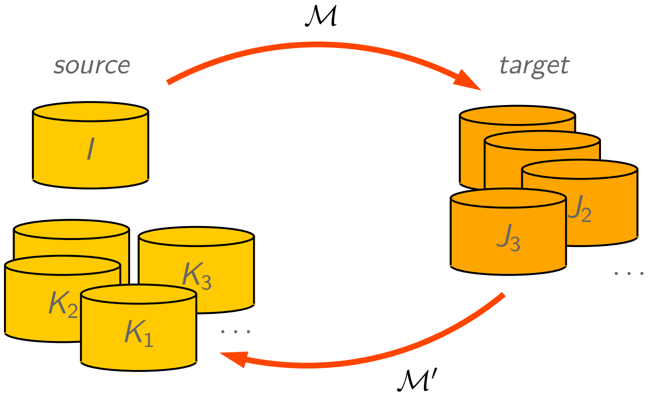
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap Q(K_2) \cap Q(K_3) \cap \cdots \quad \subseteq$$

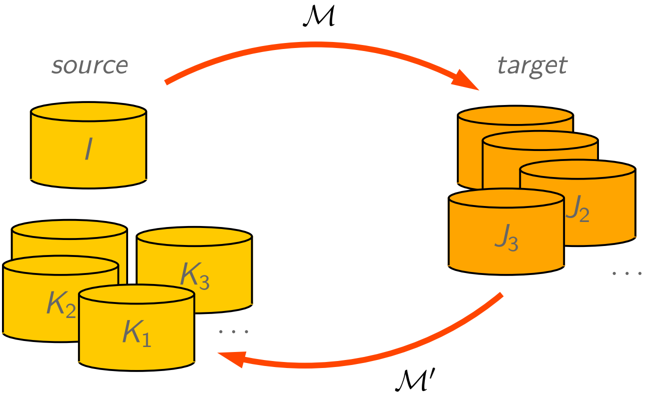
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$Q(K_1) \cap Q(K_2) \cap Q(K_3) \cap \cdots \quad \subseteq \quad Q(I)$$

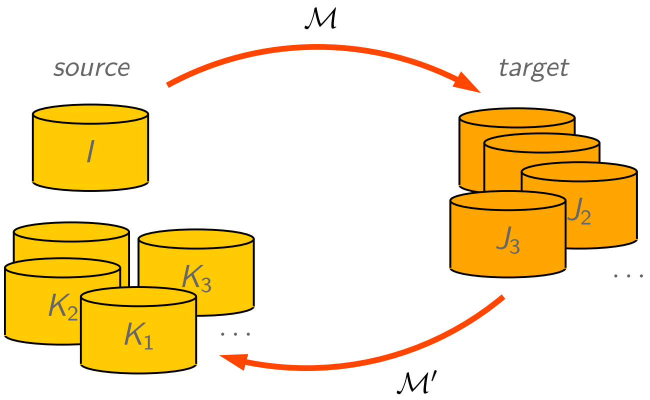
Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$\underbrace{Q(K_1) \cap Q(K_2) \cap Q(K_3) \cap \cdots}_{\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q, I)} \quad \subseteq \quad Q(I)$$

Recovering sound information wrt a query

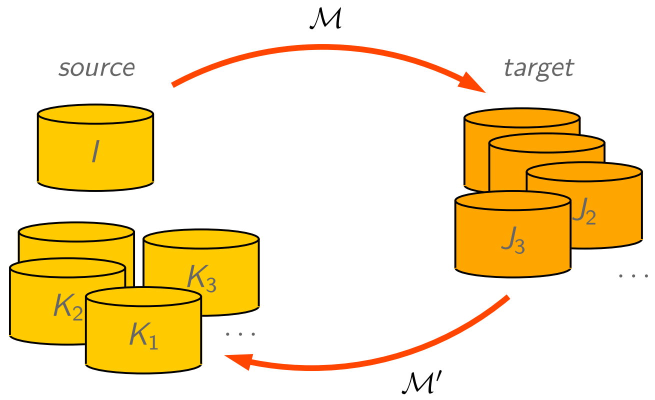


$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$\underbrace{Q(K_1) \cap Q(K_2) \cap Q(K_3) \cap \cdots}_{\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q, I)} \quad \subseteq \quad Q(I)$$

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q, I) \quad \subseteq \quad Q(I)$$

Recovering sound information wrt a query



$\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ wrt a query Q if

$$\begin{array}{ccc} \underbrace{Q(K_1) \cap Q(K_2) \cap Q(K_3) \cap \cdots}_{\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q,\,I)} & \subseteq & Q(I) \\ & \subseteq & Q(I) \end{array}$$

for every source instance I.

Recovering sound information: example

$$\mathcal{M} \quad : \quad A(x, y, z) \ \rightarrow \ \exists U \ P(x, y, U)$$

Recovering sound information: example

$$\begin{array}{rcl} \mathcal{M} & : & A(x,y,z) \ \rightarrow \ \exists U \ P(x,y,U) \\ \mathcal{M}' & : & P(x,y,u) \ \rightarrow \ A(x,y,x) \end{array}$$

Recovering sound information: example

$$
\begin{aligned}
\mathcal{M} &: A(x, y, z) \rightarrow \exists U\ P(x, y, U) \\
\mathcal{M}' &: P(x, y, u) \rightarrow A(x, y, x)
\end{aligned}
$$

Recovering sound information: example

$$\begin{aligned} \mathcal{M} &: A(x,y,z) \rightarrow \exists U \; P(x,y,U) \\ \mathcal{M}' &: P(x,y,u) \rightarrow A(x,y,x) \end{aligned}$$

$$Q_1(x) : \exists U \exists V \; A(x,U,V)$$

Recovering sound information: example

$$\begin{array}{rcl} \mathcal{M} & : & A(x,y,z) \rightarrow \exists U\ P(x,y,U) \\ \mathcal{M}' & : & P(x,y,u) \rightarrow A(x,y,x) \end{array}$$

$$Q_1(x) \ : \ \exists U \exists V\ A(x,U,V)$$

- $\mathcal{M}'$ is a Q_1-recovery of $\mathcal{M}$

Recovering sound information: example

$$\begin{aligned}\mathcal{M} &: A(x,y,z) \rightarrow \exists U\ P(x,y,U)\\ \mathcal{M}' &: P(x,y,u) \rightarrow A(x,y,x)\end{aligned}$$

$$\begin{aligned}Q_1(x) &: \exists U \exists V\ A(x,U,V)\\ Q_2(z) &: \exists U \exists V\ A(U,V,z)\end{aligned}$$

- $\mathcal{M}'$ is a Q_1-recovery of $\mathcal{M}$

Recovering sound information: example

$$\begin{array}{rcl} \mathcal{M} & : & A(x, y, z) \rightarrow \exists U \; P(x, y, U) \\ \mathcal{M}' & : & P(x, y, u) \rightarrow A(x, y, x) \end{array}$$

$$\begin{array}{rcl} Q_1(x) & : & \exists U \exists V \; A(x, U, V) \\ Q_2(z) & : & \exists U \exists V \; A(U, V, z) \end{array}$$

- $\mathcal{M}'$ is a Q_1-recovery of $\mathcal{M}$
- $\mathcal{M}'$ *is not* a Q_2-recovery of $\mathcal{M}$

Recovering sound information: example

$$\begin{array}{rcl} \mathcal{M} & : & A(x,y,z) \rightarrow \exists U\ P(x,y,U) \\ \mathcal{M}' & : & P(x,y,u) \rightarrow A(x,y,x) \end{array}$$

$$\begin{array}{rcl} Q_1(x) & : & \exists U \exists V\ A(x,U,V) \\ Q_2(z) & : & \exists U \exists V\ A(U,V,z) \end{array}$$

- $\mathcal{M}'$ is a Q_1-recovery of $\mathcal{M}$
- $\mathcal{M}'$ *is not* a Q_2-recovery of $\mathcal{M}$
  - $I = \{A(1,2,3)\}$

Recovering sound information: example

$$\mathcal{M} \quad : \quad A(x,y,z) \rightarrow \exists U \; P(x,y,U)$$
$$\mathcal{M}' \quad : \quad P(x,y,u) \rightarrow A(x,y,x)$$

$$Q_1(x) \quad : \quad \exists U \exists V \; A(x,U,V)$$
$$Q_2(z) \quad : \quad \exists U \exists V \; A(U,V,z)$$

- $\mathcal{M}'$ is a Q_1-recovery of $\mathcal{M}$
- $\mathcal{M}'$ *is not* a Q_2-recovery of $\mathcal{M}$
  - $I = \{A(1,2,3)\}$
  - $\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q_2, I) = \{1\}$

Recovering sound information: example

$$\begin{array}{rcl} \mathcal{M} & : & A(x,y,z) \rightarrow \exists U\ P(x,y,U) \\ \mathcal{M}' & : & P(x,y,u) \rightarrow A(x,y,x) \end{array}$$

$$\begin{array}{rcl} Q_1(x) & : & \exists U \exists V\ A(x,U,V) \\ Q_2(z) & : & \exists U \exists V\ A(U,V,z) \end{array}$$

- $\mathcal{M}'$ is a Q_1-recovery of $\mathcal{M}$
- $\mathcal{M}'$ *is not* a Q_2-recovery of $\mathcal{M}$
  - $I = \{A(1,2,3)\}$
  - $\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q_2, I) = \{1\}$
  - $Q_2(I) = \{3\}$

Recovering sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}'$ is a *$\mathcal{C}$-recovery* of $\mathcal{M}$ if

Recovering sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}'$ is a *$\mathcal{C}$-recovery* of $\mathcal{M}$ if

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q, I) \subseteq Q(I)$$

Recovering sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}'$ is a *$\mathcal{C}$-recovery* of $\mathcal{M}$ if

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}'}(Q, I) \subseteq Q(I)$$

for every source instance I and *for every query* $Q \in \mathcal{C}$.

Recovering sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}'$ is a *$\mathcal{C}$-recovery* of $\mathcal{M}$ if

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) \subseteq Q(I)$$

for every source instance I and *for every query* $Q \in \mathcal{C}$.

For example:

- **All**-recovery: sound information for *all possible queries*
- **CQ**-recovery: sound information for *all conjunctive queries*

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_2}(Q, I)$$

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \subseteq \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I)$$

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \subseteq \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I) \subseteq Q(I)$$

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \ \subseteq \ \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I) \ \subseteq \ Q(I)$$

for every I and $Q \in \mathcal{C}$, then

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \ \subseteq \ \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I) \ \subseteq \ Q(I)$$

for every I and $Q \in \mathcal{C}$, then

$\mathcal{M}_1$ is a *better* that $\mathcal{M}_2$ as a $\mathcal{C}$-recovery of $\mathcal{M}$.

Comparing $\mathcal{C}$-recoveries

Assume that we have two $\mathcal{C}$-recoveries $\mathcal{M}_1$ and $\mathcal{M}_2$ such that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \subseteq \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I) \subseteq Q(I)$$

for every I and $Q \in \mathcal{C}$, then

$\mathcal{M}_1$ is a *better* that $\mathcal{M}_2$ as a $\mathcal{C}$-recovery of $\mathcal{M}$.

We want a mapping such that the certain answers are *as close as possible* to $Q(I)$.

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \mathcal{C}\textit{-maximum recovery} \text{ of } \mathcal{M}$$

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \mathcal{C}\textit{-maximum recovery} \text{ of } \mathcal{M}$$

if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \textit{$\mathcal{C}$-maximum recovery} \text{ of } \mathcal{M}$$

if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_2}(Q, I)$$

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \mathcal{C}\textit{-maximum recovery} \text{ of } \mathcal{M}$$

if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_2}(Q, I) \ \subseteq \ \underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_1}(Q, I)$$

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \textit{$\mathcal{C}$-maximum recovery} \text{ of } \mathcal{M}$$

if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \subseteq \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I) \subseteq Q(I)$$

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \mathcal{C}\textit{-maximum recovery} \text{ of } \mathcal{M}$$

if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_2}(Q, I) \ \subseteq \ \underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_1}(Q, I) \ \subseteq \ Q(I)$$

for every source instance I and for every query $Q \in \mathcal{C}$.

Recovering the maximum amount of sound information wrt a class of queries

Definition

Given a class of queries $\mathcal{C}$, we say that

$$\mathcal{M}_1 \text{ is a } \mathcal{C}\textit{-maximum recovery} \text{ of } \mathcal{M}$$

if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

$$\underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_2}(Q, I) \subseteq \underline{\text{certain}}_{\mathcal{M} \circ \mathcal{M}_1}(Q, I) \subseteq Q(I)$$

for every source instance I and for every query $Q \in \mathcal{C}$.

$\mathcal{M}_1$ is better that any other possible $\mathcal{C}$-recovery!

Previous notions of inverse correspond to particular classes of queries

Let $\mathcal{M}$ be specified by tgds:

Previous notions of inverse correspond to particular classes of queries

Let $\mathcal{M}$ be specified by tgds:

Theorem

- *If $\mathcal{M}$ has a Fagin-inverse, then:*

  $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ iff
  $\mathcal{M}'$ is a $\mathbf{UCQ}^{\neq}$-maximum recovery of $\mathcal{M}$.

Previous notions of inverse correspond to particular classes of queries

Let $\mathcal{M}$ be specified by tgds:

Theorem

- *If $\mathcal{M}$ has a Fagin-inverse, then:*

$$\mathcal{M}' \text{ is a Fagin-inverse of } \mathcal{M} \text{ iff}$$
$$\mathcal{M}' \text{ is a } \mathbf{UCQ}^{\neq}\text{-maximum recovery of } \mathcal{M}.$$

Theorem

- *If $\mathcal{M}$ has a quasi-inverse, then there exists a class of queries $\mathcal{C} \subseteq \mathbf{UCQ}^{\neq}$ such that:*

$$\mathcal{M}' \text{ is a quasi-inverse of } \mathcal{M} \text{ iff}$$
$$\mathcal{M}' \text{ is a } \mathcal{C}\text{-maximum recovery of } \mathcal{M}.$$

Previous notions of inverse correspond to particular classes of queries

Let $\mathcal{M}$ be specified by tgds:

Theorem

- *If $\mathcal{M}$ has a Fagin-inverse, then:*

$$\mathcal{M}' \text{ is a Fagin-inverse of } \mathcal{M} \text{ iff}$$
$$\mathcal{M}' \text{ is a } \mathbf{UCQ}^{\neq}\text{-maximum recovery of } \mathcal{M}.$$

Theorem

- *If $\mathcal{M}$ has a quasi-inverse, then there exists a class of queries $\mathcal{C} \subseteq \mathbf{UCQ}^{\neq}$ such that:*

$$\mathcal{M}' \text{ is a quasi-inverse of } \mathcal{M} \text{ iff}$$
$$\mathcal{M}' \text{ is a } \mathcal{C}\text{-maximum recovery of } \mathcal{M}.$$

If $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$
then $\mathcal{M}'$ is an **All**-maximum recovery of $\mathcal{M}$.

Why do we need another notion of inverse?

Why do we need another notion of inverse?

Problem 1:

- Fagin-inverses rarely exist for tgds

Why do we need another notion of inverse?

Problem 1:

- Fagin-inverses rarely exist for tgds

Problem 2:

- Quasi-inverse and maximum recovery of tgds need disjunctions to be expressed:

  tgds with disjunctions in the right-hand side.

Why do we need another notion of inverse?

Problem 1:

- Fagin-inverses rarely exist for tgds

Problem 2:

- Quasi-inverse and maximum recovery of tgds need disjunctions to be expressed:

  tgds with disjunctions in the right-hand side.

- How do we exchange data using tgds with disjunctions?

Why do we need another notion of inverse?

Problem 1:

- Fagin-inverses rarely exist for tgds

Problem 2:

- Quasi-inverse and maximum recovery of tgds need disjunctions to be expressed:

  tgds with disjunctions in the right-hand side.

- How do we exchange data using tgds with disjunctions?

We would like a *natural* notion of inverse such that:

- tgds always have an inverse, and
- such inverse can be expressed in a language with the same good properties as tgds for data exchange.

When focusing on **CQ** these issues can be solved

Main Theorem

Every mapping specified by tgds has a **CQ**-maximum recovery that can be specified by tgds with $\neq$ and $\mathbf{C}(\cdot)$ in the left-hand side.

When focusing on **CQ** these issues can be solved

Main Theorem

Every mapping specified by tgds has a **CQ**-maximum recovery that can be specified by tgds with $\neq$ and $\mathbf{C}(\cdot)$ in the left-hand side.

Proof idea

We provide an algorithm to compute **CQ**-maximum recoveries of tgds that has as output a set of tgds$^{\neq,\mathbf{C}}$, and prove its correctness.

Query rewriting: a key concept in the algorithm

Query rewriting:

- Q' is a *rewriting* of Q under $\mathcal{M}$ if

$$\underline{\text{certain}}_{\mathcal{M}}(Q, I)$$

Query rewriting: a key concept in the algorithm

Query rewriting:

- Q' is a *rewriting* of Q under $\mathcal{M}$ if

$$\underline{\text{certain}}_{\mathcal{M}}(Q, I) = Q'(I)$$

for every I.

Query rewriting: a key concept in the algorithm

Query rewriting:

- Q' is a *rewriting* of Q under $\mathcal{M}$ if

$$\underline{\text{certain}}_{\mathcal{M}}(Q, I) = Q'(I)$$

for every I.

Well-known result:

If $\mathcal{M}$ is specified by tgds, then for every query $Q \in \mathbf{CQ}$, there exists a query $Q' \in \mathbf{UCQ}^{=}$ that is a rewriting of Q under $\mathcal{M}$.

Step 1: compute a **CQ**-maximum recovery using rewriting of queries.

For a mapping $\mathcal{M}$ specified by tgds, compute $\mathcal{M}'$ as follows:

Step 1

For every dependency $\varphi(\bar{x}) \to \exists \bar{y} \psi(\bar{x}, \bar{y})$ defining $\mathcal{M}$:

Step 1: compute a **CQ**-maximum recovery using rewriting of queries.

For a mapping $\mathcal{M}$ specified by tgds, compute $\mathcal{M}'$ as follows:

Step 1

For every dependency $\varphi(\bar{x}) \rightarrow \exists\bar{y}\psi(\bar{x}, \bar{y})$ defining $\mathcal{M}$:

- Let $\alpha(\bar{x}) \in \mathbf{UCQ}^{=}$ be a rewriting of $\exists\bar{y}\psi(\bar{x}, \bar{y})$ under $\mathcal{M}$.

Step 1: compute a **CQ**-maximum recovery using rewriting of queries.

For a mapping $\mathcal{M}$ specified by tgds, compute $\mathcal{M}'$ as follows:

Step 1

For every dependency $\varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$ defining $\mathcal{M}$:

- Let $\alpha(\bar{x}) \in \mathbf{UCQ}^{=}$ be a rewriting of $\exists \bar{y} \psi(\bar{x}, \bar{y})$ under $\mathcal{M}$.
- Add to the definition of $\mathcal{M}'$ the dependency

$$\exists \bar{y} \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \rightarrow \alpha(\bar{x}).$$

Step 1: compute a **CQ**-maximum recovery using rewriting of queries.

For a mapping $\mathcal{M}$ specified by tgds, compute $\mathcal{M}'$ as follows:

Step 1

For every dependency $\varphi(\bar{x}) \to \exists\bar{y}\psi(\bar{x},\bar{y})$ defining $\mathcal{M}$:

- Let $\alpha(\bar{x}) \in \mathbf{UCQ}^{=}$ be a rewriting of $\exists\bar{y}\psi(\bar{x},\bar{y})$ under $\mathcal{M}$.
- Add to the definition of $\mathcal{M}'$ the dependency

$$\exists\bar{y}\psi(\bar{x},\bar{y}) \land \mathbf{C}(\bar{x}) \to \alpha(\bar{x}).$$

Lemma

$\mathcal{M}'$ is a **CQ***-maximum recovery of $\mathcal{M}$.*

Step 1: compute a **CQ**-maximum recovery using rewriting of queries.

For a mapping $\mathcal{M}$ specified by tgds, compute $\mathcal{M}'$ as follows:

Step 1

For every dependency $\varphi(\bar{x}) \to \exists \bar{y} \psi(\bar{x}, \bar{y})$ defining $\mathcal{M}$:

- Let $\alpha(\bar{x}) \in \mathbf{UCQ}^{=}$ be a rewriting of $\exists \bar{y} \psi(\bar{x}, \bar{y})$ under $\mathcal{M}$.
- Add to the definition of $\mathcal{M}'$ the dependency

$$\exists \bar{y} \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \to \alpha(\bar{x}).$$

Lemma

*$\mathcal{M}'$ is a **CQ**-maximum recovery of $\mathcal{M}$.*

Problem: disjunctions and equalities in the right-hand side.

Step 2: eliminate right-hand side equalities

Example:

$$A(x,y) \quad \rightarrow \quad R(x,y) \quad \vee \quad (P(x) \wedge x = y) \quad \vee \quad S(x) \wedge T(y)$$

Step 2: eliminate right-hand side equalities

Example:

$$A(x, y) \quad \rightarrow \quad R(x, y) \quad \vee \quad (P(x) \wedge x = y) \quad \vee \quad S(x) \wedge T(y)$$

$$\Downarrow$$

Step 2: eliminate right-hand side equalities

Example:

$$
\begin{array}{ccc}
A(x, y) & \rightarrow & R(x, y) \quad \vee \quad (P(x) \wedge x = y) \quad \vee \quad S(x) \wedge T(y) \\
 & & \Downarrow \\
A(x, y) \wedge x \neq y & \rightarrow &
\end{array}
$$

Step 2: eliminate right-hand side equalities

Example:

$$
\begin{array}{rcccccc}
A(x,y) & \rightarrow & R(x,y) & \vee & (P(x) \wedge x = y) & \vee & S(x) \wedge T(y) \\
 & & & \Downarrow & & & \\
A(x,y) \wedge x \neq y & \rightarrow & R(x,y) & \vee & & \vee & S(x) \wedge T(y)
\end{array}
$$

Step 2: eliminate right-hand side equalities

Example:

$$
\begin{array}{rcccccc}
A(x,y) & \rightarrow & R(x,y) & \vee & (P(x) \wedge x = y) & \vee & S(x) \wedge T(y) \\
 & & & \Downarrow & & & \\
A(x,y) \wedge x \neq y & \rightarrow & R(x,y) & \vee & & \vee & S(x) \wedge T(y) \\
A(x,x) & \rightarrow & & & & &
\end{array}
$$

Step 2: eliminate right-hand side equalities

Example:

$$
\begin{array}{ccccccc}
A(x,y) & \rightarrow & R(x,y) & \vee & (P(x) \wedge x = y) & \vee & S(x) \wedge T(y) \\
 & & & \Downarrow & & & \\
A(x,y) \wedge x \neq y & \rightarrow & R(x,y) & \vee & & \vee & S(x) \wedge T(y) \\
A(x,x) & \rightarrow & R(x,x) & \vee & P(x) & \vee & S(x) \wedge T(x)
\end{array}
$$

Step 2: eliminate right-hand side equalities

Example:

$$
\begin{array}{ccccccc}
A(x,y) & \to & R(x,y) & \vee & (P(x) \wedge x = y) & \vee & S(x) \wedge T(y) \\
 & & & \Downarrow & & & \\
A(x,y) \wedge x \neq y & \to & R(x,y) & \vee & & \vee & S(x) \wedge T(y) \\
A(x,x) & \to & R(x,x) & \vee & P(x) & \vee & S(x) \wedge T(x)
\end{array}
$$

Step 2

- Let $\mathcal{M}'$ be the mapping constructed in Step 1.
- Construct $\mathcal{M}''$ from $\mathcal{M}'$ by replacing right-hand side equalities by inequalities in the left-hand side.

Step 2: eliminate right-hand side equalities

Example:

$$\begin{array}{rcccccc}
A(x,y) & \rightarrow & R(x,y) & \vee & (P(x) \wedge x = y) & \vee & S(x) \wedge T(y) \\
 & & & \Downarrow & & & \\
A(x,y) \wedge x \neq y & \rightarrow & R(x,y) & \vee & & \vee & S(x) \wedge T(y) \\
A(x,x) & \rightarrow & R(x,x) & \vee & P(x) & \vee & S(x) \wedge T(x)
\end{array}$$

Step 2

- Let $\mathcal{M}'$ be the mapping constructed in Step 1.
- Construct $\mathcal{M}''$ from $\mathcal{M}'$ by replacing right-hand side equalities by inequalities in the left-hand side.

Lemma

*$\mathcal{M}''$ is a **CQ**-maximum recovery of $\mathcal{M}$.*

Step 2: eliminate right-hand side equalities

Example:

$$
\begin{array}{rcccccc}
A(x,y) & \rightarrow & R(x,y) & \vee & (P(x) \wedge x = y) & \vee & S(x) \wedge T(y) \\
 & & & \Downarrow & & & \\
A(x,y) \wedge x \neq y & \rightarrow & R(x,y) & \vee & & \vee & S(x) \wedge T(y) \\
A(x,x) & \rightarrow & R(x,x) & \vee & P(x) & \vee & S(x) \wedge T(x)
\end{array}
$$

Step 2

- Let $\mathcal{M}'$ be the mapping constructed in Step 1.
- Construct $\mathcal{M}''$ from $\mathcal{M}'$ by replacing right-hand side equalities by inequalities in the left-hand side.

Lemma

$\mathcal{M}''$ is a **CQ***-maximum recovery of $\mathcal{M}$.*

Problem: formulas still have disjunctions in the right-hand side.

Key concept in Step 3:
Cartesian product of queries (intuition)

$$\begin{array}{rcr} Q_1(x_1, x_2) & : & T(x_1, x_2) \wedge R(x_1, x_1) \\ Q_2(x_1, x_2) & : & \exists y \;\; T(x_1, y) \wedge R(x_2, x_2) \end{array}$$

Key concept in Step 3: *Cartesian product* of queries (intuition)

$$\begin{array}{lcl} Q_1(x_1, x_2) & : & T(x_1, x_2) \wedge R(x_1, x_1) \\ Q_2(x_1, x_2) & : & \exists y \; T(x_1, y) \wedge R(x_2, x_2) \end{array}$$

If we know that tuple (a, b) is an answer to one of the two queries

What can we *certainly infer* about tables T and R?

Key concept in Step 3: *Cartesian product* of queries (intuition)

$$\begin{array}{rcl} Q_1(x_1, x_2) & : & T(x_1, x_2) \wedge R(x_1, x_1) \\ Q_2(x_1, x_2) & : & \exists y \; T(x_1, y) \wedge R(x_2, x_2) \end{array}$$

If we know that tuple (a, b) is an answer to one of the two queries

What can we *certainly infer* about tables T and R?

- element a is in the first component of T

Key concept in Step 3: *Cartesian product* of queries (intuition)

$$\begin{aligned} Q_1(x_1, x_2) &: \quad T(x_1, x_2) \wedge R(x_1, x_1) \\ Q_2(x_1, x_2) &: \quad \exists y \; T(x_1, y) \wedge R(x_2, x_2) \end{aligned}$$

If we know that tuple (a, b) is an answer to one of the two queries

What can we *certainly infer* about tables T and R?

- element a is in the first component of T
- there is *some element* in both components of R

Key concept in Step 3:
Cartesian product of queries (intuition)

$$
\begin{array}{rcl}
Q_1(x_1, x_2) & : & T(x_1, x_2) \wedge R(x_1, x_1) \\
Q_2(x_1, x_2) & : & \exists y \; T(x_1, y) \wedge R(x_2, x_2)
\end{array}
$$

If we know that tuple (a, b) is an answer to one of the two queries

What can we *certainly infer* about tables T and R?

- element a is in the first component of T
- there is *some element* in both components of R

$$Q(x_1) \; : \; \exists u \exists v \; T(x_1, u) \wedge R(v, v)$$

Key concept in Step 3:
Cartesian product of queries (intuition)

$$
\begin{array}{lcr}
Q_1(x_1, x_2) & : & T(x_1, x_2) \wedge R(x_1, x_1) \\
Q_2(x_1, x_2) & : & \exists y \ T(x_1, y) \wedge R(x_2, x_2)
\end{array}
$$

If we know that tuple (a, b) is an answer to one of the two queries

What can we *certainly infer* about tables T and R?

- element a is in the first component of T
- there is *some element* in both components of R

$$Q(x_1) \ : \ \exists u \exists v \ T(x_1, u) \wedge R(v, v)$$

$Q(x_1)$ is the *Cartesian product* of $Q_1(x_1, x_2)$ and $Q_2(x_1, x_2)$.

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition

Homomorphism between conjunctive queries: function h that

- maps existential variables to free or existential variables
- is the identity over free variables

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition

Homomorphism between conjunctive queries: function h that

- maps existential variables to free or existential variables
- is the identity over free variables

$$\exists u \exists v \;\; T(x_1, u) \wedge R(v, v) \qquad \xrightarrow{h_1} \qquad T(x_1, x_2) \wedge R(x_1, x_1)$$

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition

Homomorphism between conjunctive queries: function h that

- maps existential variables to free or existential variables
- is the identity over free variables

$$\exists u \exists v \; T(x_1, u) \wedge R(v, v) \quad \xrightarrow[h_1(u) = x_2]{h_1} \quad T(x_1, x_2) \wedge R(x_1, x_1)$$

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition

Homomorphism between conjunctive queries: function h that

- maps existential variables to free or existential variables
- is the identity over free variables

$$\exists u \exists v \; T(x_1, u) \land R(v, v) \quad \xrightarrow{h_1} \quad T(x_1, x_2) \land R(x_1, x_1)$$

$$h_1(u) = x_2$$
$$h_1(v) = x_1$$

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$\begin{array}{ccc} Q_1 & & Q_2 \\ & Q & \end{array}$$

Key concept in Step 3:
Cartesian product of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$\begin{tikzcd} Q_1 & & Q_2 \\ & Q \arrow[ul, "h_1"] \arrow[ur, "h_2"'] & \end{tikzcd}$$

Key concept in Step 3:
Cartesian product of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$\begin{tikzcd}
Q_1 & & Q_2 \\
& Q \arrow[ul, "h_1"'] \arrow[ur, "h_2"] & \\
& Q' &
\end{tikzcd}$$

Key concept in Step 3:
Cartesian product of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$\begin{tikzcd}
Q_1 & & Q_2 \\
& Q \arrow[ul, "h_1"'] \arrow[ur, "h_2"] & \\
& Q' \arrow[uul, "h_1'"] \arrow[uur, "h_2'"'] &
\end{tikzcd}$$

Key concept in Step 3:
Cartesian product of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$\begin{tikzcd}
Q_1 & & Q_2 \\
& Q \arrow[ul, "h_1"'] \arrow[ur, "h_2"] & \\
& Q' \arrow[uul, "h_1'"] \arrow[u, "h'"'] \arrow[uur, "h_2'"'] &
\end{tikzcd}$$

Key concept in Step 3: *Cartesian product* of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$
\begin{tikzcd}
Q_1 & & Q_2 \\
& Q \arrow[ul, "h_1"'] \arrow[ur, "h_2"] & \\
& Q' \arrow[uul, "h'_1"] \arrow[uur, "h'_2"'] \arrow[u, "h'"'] &
\end{tikzcd}
\qquad Q = Q_1 \times Q_2
$$

Key concept in Step 3:
Cartesian product of queries (formalization)

Definition (semantic version)

The conjuntive query Q is the *Cartesian product* of Q_1 and Q_2 if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

$$\begin{tikzcd}
Q_1 & & Q_2 \\
& Q \arrow[ul, "h_1"'] \arrow[ur, "h_2"] & \\
& Q' \arrow[u, "h'"'] \arrow[uul, "h'_1"] \arrow[uur, "h'_2"'] &
\end{tikzcd}
\qquad Q = Q_1 \times Q_2$$

In the paper we give an algorithm for the Cartesian product:

- a simple extension of the Cartesian product of graphs.

Step 3: eliminate disjunctions

Step 3

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.
- Construct $\mathcal{M}^\star$ by replacing every dependency

$$\varphi(\bar{x}) \;\rightarrow\; \beta_1(\bar{x}) \;\vee\; \beta_2(\bar{x}) \;\vee\; \cdots \;\vee\; \beta_n(\bar{x})$$

  by

Step 3: eliminate disjunctions

Step 3

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.
- Construct $\mathcal{M}^\star$ by replacing every dependency

$$\varphi(\bar{x}) \; \rightarrow \; \beta_1(\bar{x}) \; \vee \; \beta_2(\bar{x}) \; \vee \; \cdots \; \vee \; \beta_n(\bar{x})$$

  by

$$\varphi(\bar{x}) \; \rightarrow \; \beta_1(\bar{x}) \; \times \; \beta_2(\bar{x}) \; \times \; \cdots \; \times \; \beta_n(\bar{x})$$

Step 3: eliminate disjunctions

Step 3

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.
- Construct $\mathcal{M}^\star$ by replacing every dependency

$$\varphi(\bar{x}) \rightarrow \beta_1(\bar{x}) \vee \beta_2(\bar{x}) \vee \cdots \vee \beta_n(\bar{x})$$

  by

$$\varphi(\bar{x}) \rightarrow \beta_1(\bar{x}) \times \beta_2(\bar{x}) \times \cdots \times \beta_n(\bar{x})$$

Lemma

$\mathcal{M}^\star$ is a **CQ***-maximum recovery of $\mathcal{M}$.*

Summing up...

Algorithm

Let $\mathcal{M}$ be a mapping specified by tgds:

1. Compute a **CQ**-maximum recovery by using rewriting.
2. Eliminate equalities using inequalities in the left-hand side.
3. Eliminate disjunctions using Cartesian product of queries.

Summing up...

Algorithm

Let $\mathcal{M}$ be a mapping specified by tgds:

1. Compute a **CQ**-maximum recovery by using rewriting.
2. Eliminate equalities using inequalities in the left-hand side.
3. Eliminate disjunctions using Cartesian product of queries.

The mapping $\mathcal{M}^\star$ returned by the algorithm is a **CQ**-maximum recovery of $\mathcal{M}$ specified by $\text{tgds}^{\neq,\mathbf{C}}$.

Summing up...

Algorithm

Let $\mathcal{M}$ be a mapping specified by tgds:

1. Compute a **CQ**-maximum recovery by using rewriting.
2. Eliminate equalities using inequalities in the left-hand side.
3. Eliminate disjunctions using Cartesian product of queries.

The mapping $\mathcal{M}^\star$ returned by the algorithm is a **CQ**-maximum recovery of $\mathcal{M}$ specified by $\text{tgds}^{\neq,\mathbf{C}}$.

Highlights of the algorithm:

- We use query-rewriting to compute **CQ**-maximum recoveries:

Summing up...

Algorithm

Let $\mathcal{M}$ be a mapping specified by tgds:

1. Compute a **CQ**-maximum recovery by using rewriting.
2. Eliminate equalities using inequalities in the left-hand side.
3. Eliminate disjunctions using Cartesian product of queries.

The mapping $\mathcal{M}^\star$ returned by the algorithm is a **CQ**-maximum recovery of $\mathcal{M}$ specified by $\text{tgds}^{\neq,\mathbf{C}}$.

Highlights of the algorithm:

- We use query-rewriting to compute **CQ**-maximum recoveries:
  - exponential-time worst case, but

Summing up...

Algorithm

Let $\mathcal{M}$ be a mapping specified by tgds:

1. Compute a **CQ**-maximum recovery by using rewriting.
2. Eliminate equalities using inequalities in the left-hand side.
3. Eliminate disjunctions using Cartesian product of queries.

The mapping $\mathcal{M}^\star$ returned by the algorithm is a **CQ**-maximum recovery of $\mathcal{M}$ specified by tgds$^{\neq,\mathbf{C}}$.

Highlights of the algorithm:

- We use query-rewriting to compute **CQ**-maximum recoveries:
  - exponential-time worst case, but
  - we can reuse the large body of work on query-rewriting.

The language of **CQ**-maximum recoveries

Theorem

The language of tgds$^{\neq,\mathbf{C}}$ *is the* minimal language *needed to specify* **CQ***-maximum recoveries of tgds.*

The language of **CQ**-maximum recoveries

Theorem

The language of tgds$^{\neq,\mathbf{C}}$ *is the* minimal language *needed to specify* **CQ**-*maximum recoveries of tgds.*

The language has the same good properties as tgds, in particular:

- the *chase* procedure can be used to exchange data,
- a *canonical universal solution* (and a *core*) exists for every source instance.

Concluding remarks

- A new notion of inverse of schema mappings based on queries and certain answers
- Previously proposed notions of inverse can be obtained by considering specific query languages.
- When focusing on **CQ** some practical issues are solved, in particular:

  Every mapping specified by tgds has a
  CQ-maximum recovery specified in a language
  with the same good properties as tgds for data exchange.

Inverting Schema Mappings: Bridging the Gap between Theory and Practice

Marcelo Arenas, Jorge Pérez, Cristian Riveros, Juan Reutter

Computer Science Department, PUC – Chile