Recovering information in data exchange

Cristian Riveros
Khipu Institute

Oxford University
Tue 26 Jan 2010

Data exchange and integration are one of the most important data management tasks today

Data exchange and integration are one of the most important data management tasks today

- 68% of the CEOs consider the integration of distinct applications as a *key issue* for their business.

Data exchange and integration are one of the most important data management tasks today

- 68% of the CEOs consider the integration of distinct applications as a *key issue* for their business.
- 50% of total budget in IT projects are spent on integrating applications.

Data exchange and integration are one of the most important data management tasks today

- 68% of the CEOs consider the integration of distinct applications as a *key issue* for their business.
- 50% of total budget in IT projects are spent on integrating applications.
- 40% of the effort in a software project is spent in mapping similar data.

Data exchange and integration are one of the most important data management tasks today

- 68% of the CEOs consider the integration of distinct applications as a *key issue* for their business.
- 50% of total budget in IT projects are spent on integrating applications.
- 40% of the effort in a software project is spent in mapping similar data.

Data exchange/integration are crucial for achieving interoperability of applications.

Schema mappings are essential to perform the exchange of data

Schema mappings are essential to perform the exchange of data

Schema mapping describes the relationship between schemas.

Schema mappings are essential to perform the exchange of data

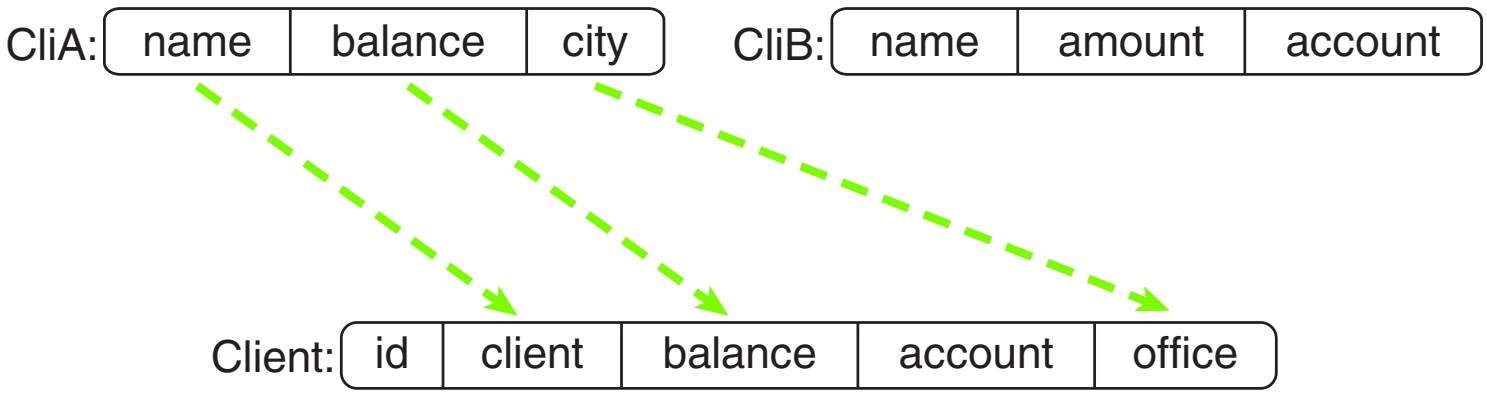
Schema mapping describes the relationship between schemas.

Schema mappings are essential to perform the exchange of data

Schema mapping describes the relationship between schemas.

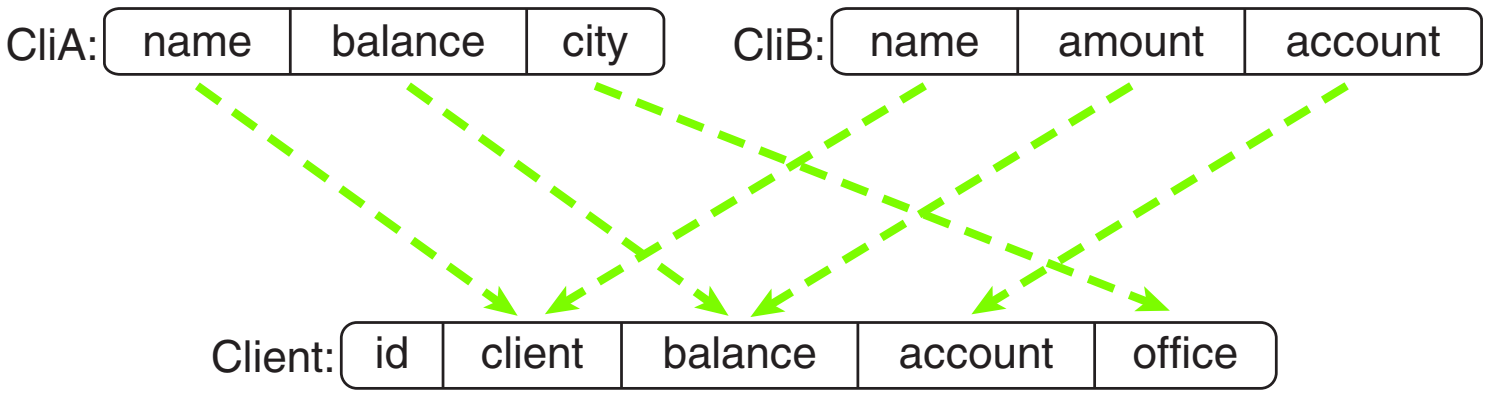
Schema mappings are essential to perform the exchange of data

Schema mapping describes the relationship between schemas.



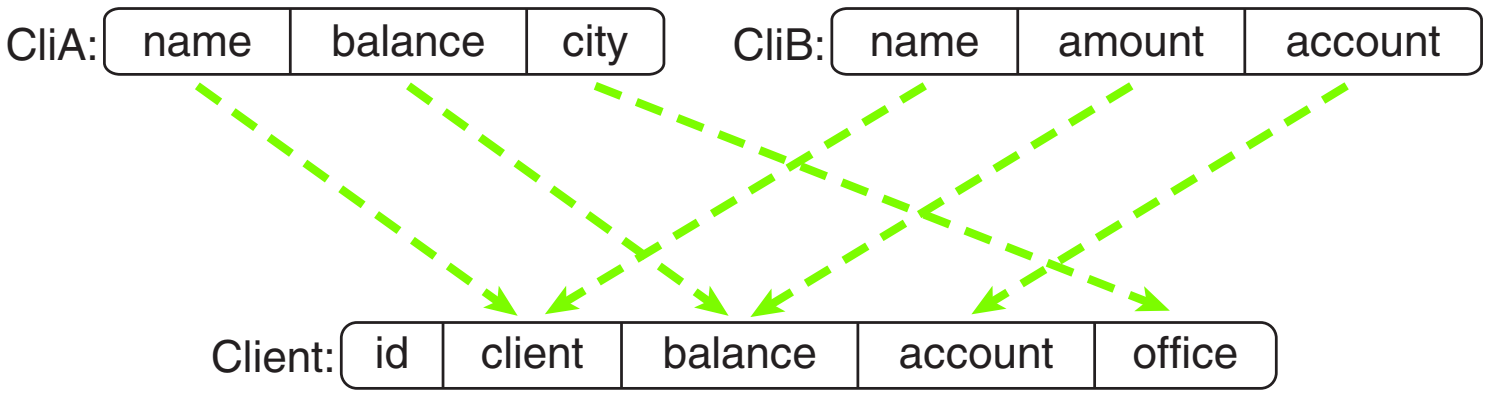
Schema mappings are essential to perform the exchange of data

Schema mapping describes the relationship between schemas.



Schema mappings are essential to perform the exchange of data

Schema mapping describes the relationship between schemas.



Schema mappings contain metadata.

In several applications we need

to reuse the metadata of schema mappings

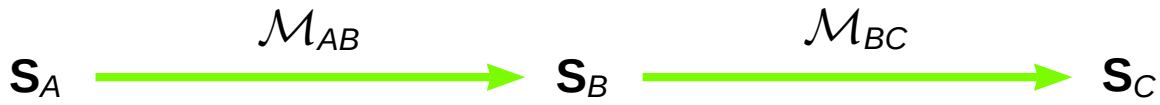
In several applications we need

to reuse the metadata of schema mappings

$\mathbf{S}_A$ $\mathbf{S}_B$ $\mathbf{S}_C$

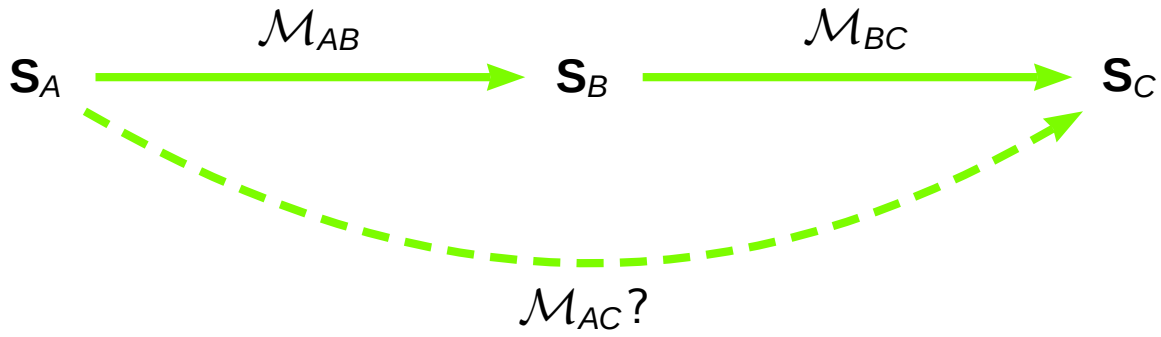
In several applications we need

to reuse the metadata of schema mappings



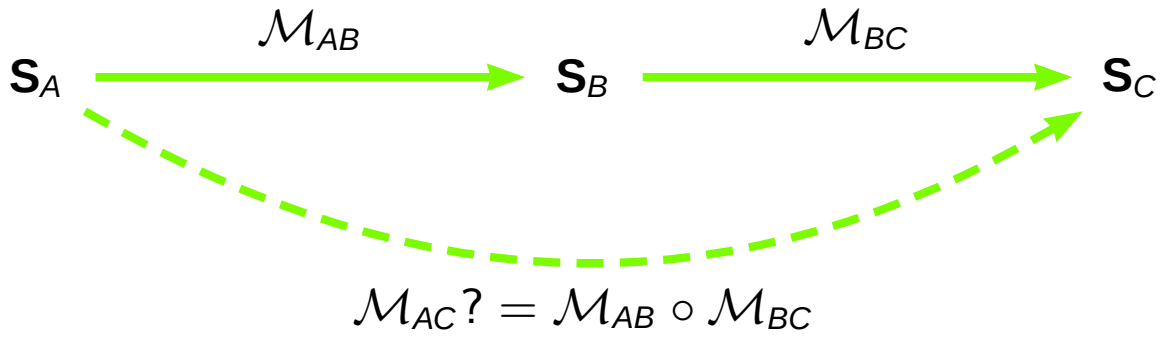
In several applications we need

to reuse the metadata of schema mappings



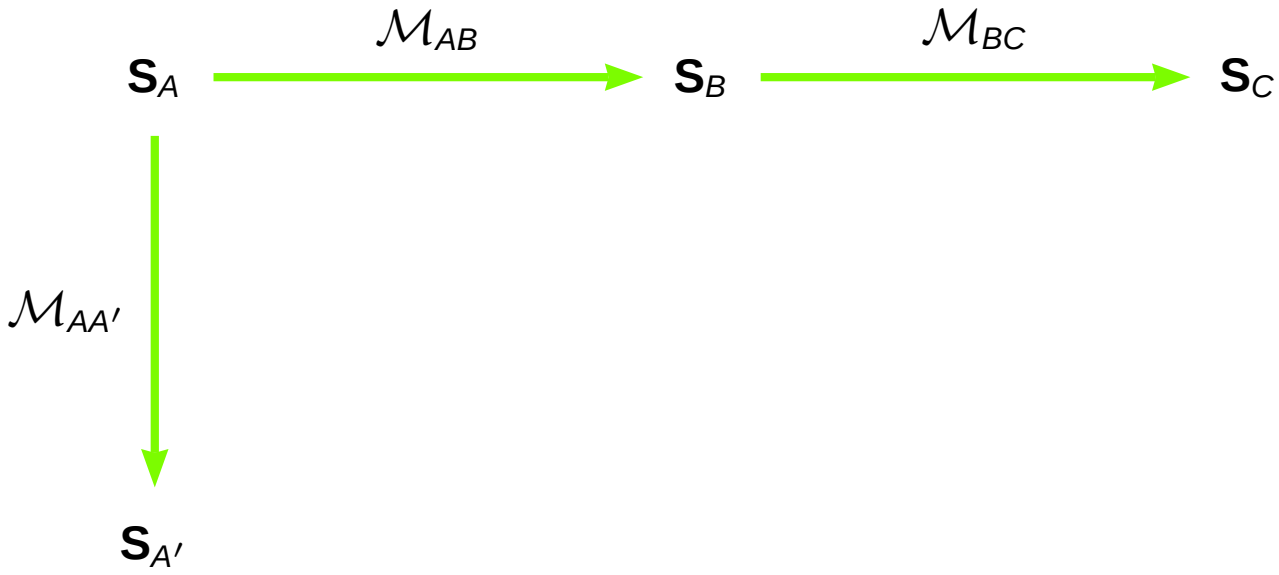
In several applications we need

to reuse the metadata of schema mappings



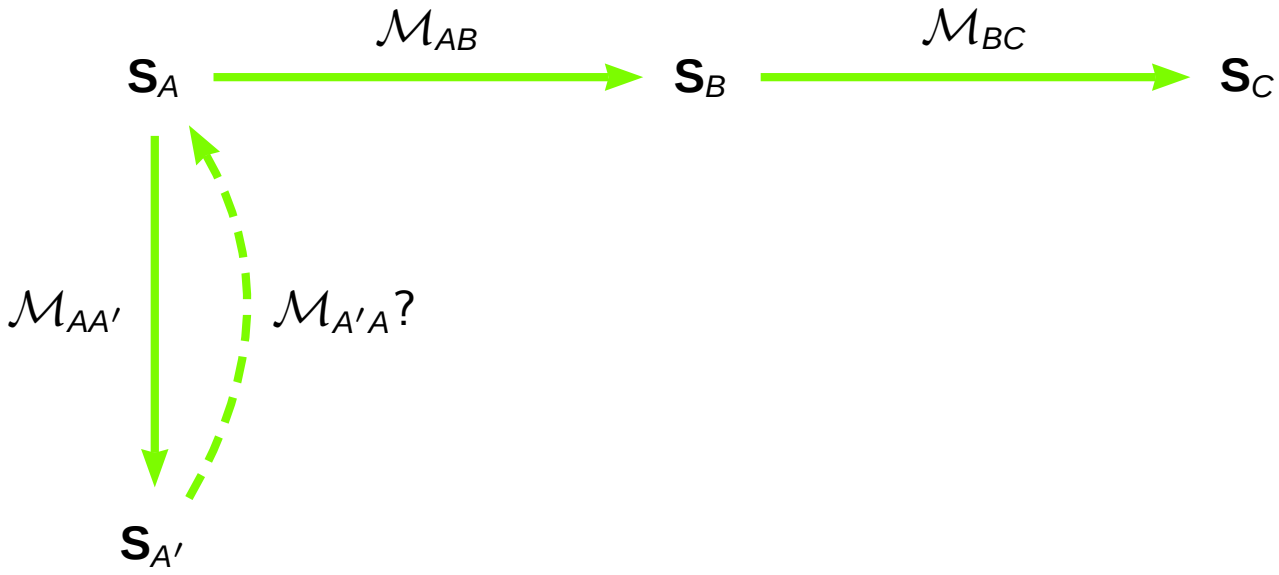
In several applications we need

to reuse the metadata of schema mappings



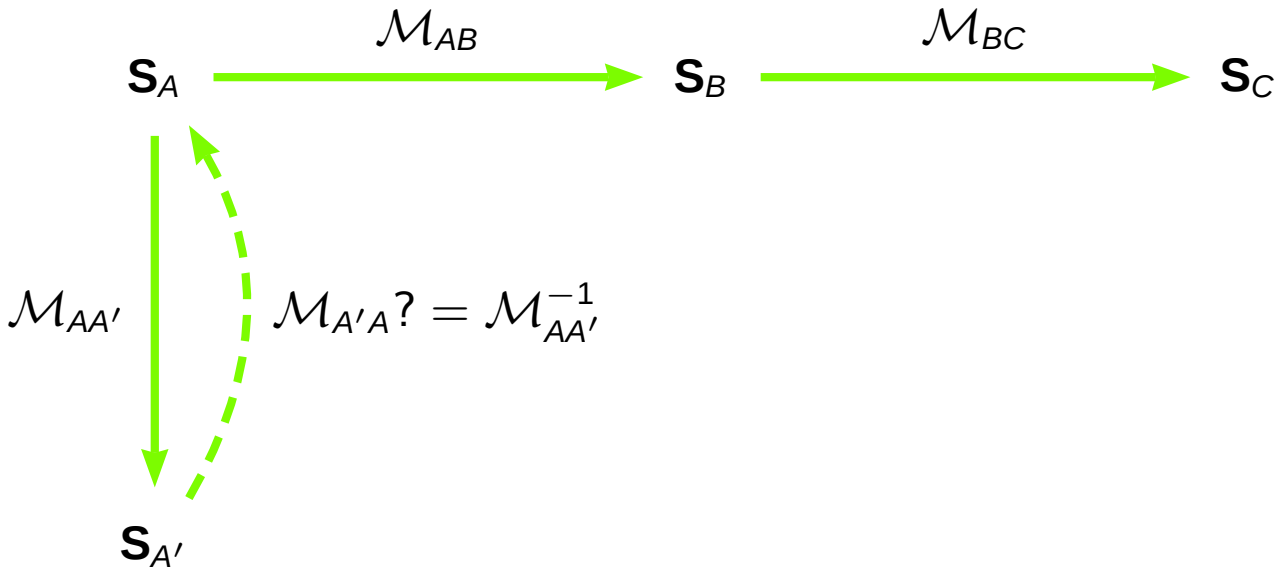
In several applications we need

to reuse the metadata of schema mappings



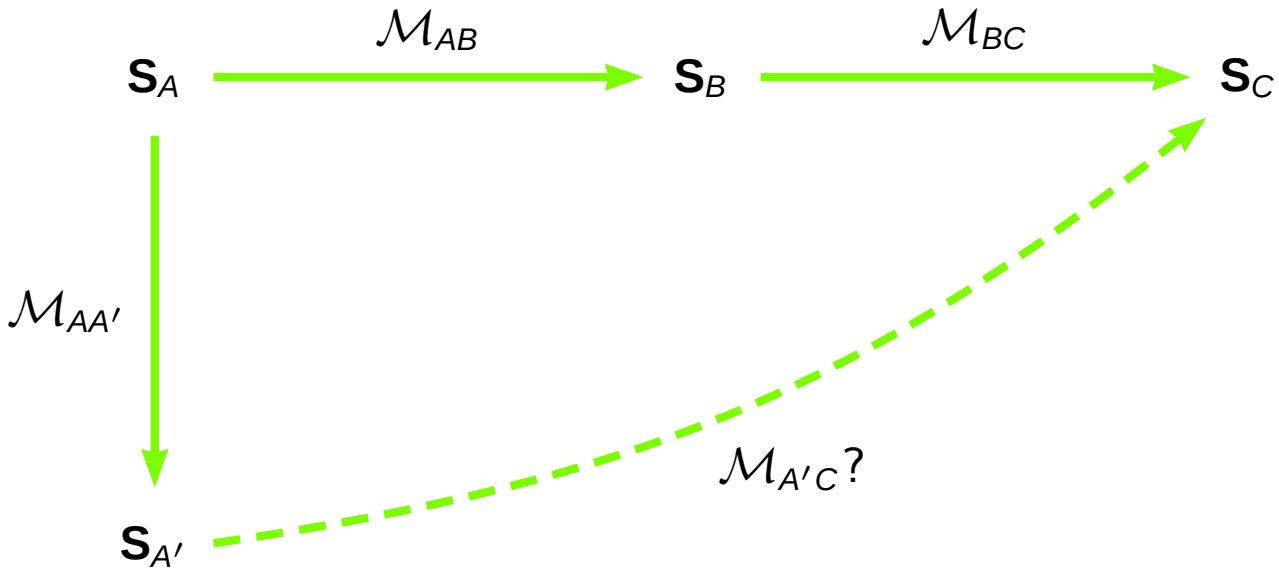
In several applications we need

to reuse the metadata of schema mappings

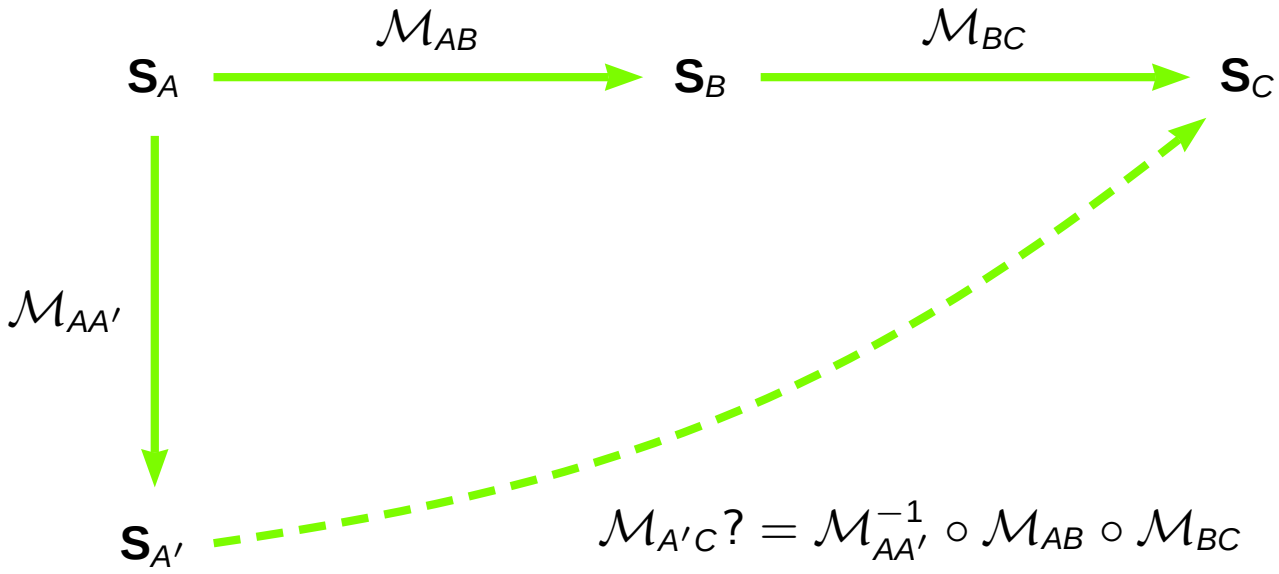


In several applications we need

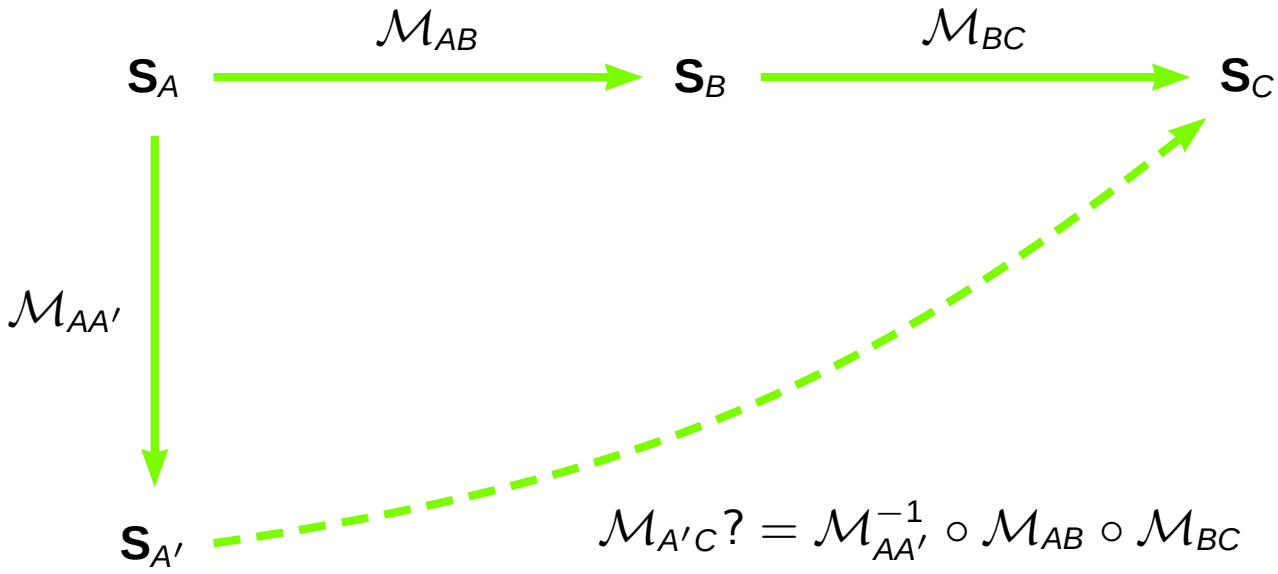
to reuse the metadata of schema mappings



In several applications we need to reuse the metadata of schema mappings



In several applications we need to reuse the metadata of schema mappings



"The goal is to develop a model management engine that can support tools for all of these applications."

Phil Bernstein, Microsoft Research

In recent years, there has been an increasing interest to develop solid foundations for metadata management

In recent years, there has been an increasing interest to develop solid foundations for metadata management

2003 *Applying model management to classical meta data problems.*
P. Bernstein. *CIDR.*

2005 *Composing schema mappings: second-order dependencies . . .*
R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. *PODS.*

2006 *Inverting schema mappings.*
R. Fagin. *PODS.*

2007 *Quasi-inverses of schema mappings*
R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. *PODS.*

2008 *The recovery of a schema mapping: bringing exchanged . . .*
M. Arenas, J. Pérez, and C. Riveros. *PODS.*

2009 *Inverting schema mappings: bridging the gap . . .*
M. Arenas, J. Pérez, J. Reutter, and C. Riveros. *VLDB.*

In recent years, there has been an increasing interest to develop solid foundations for metadata management

2003 *Applying model management to classical meta data problems.*
P. Bernstein. *CIDR.*

2005 *Composing schema mappings: second-order dependencies . . .*
R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. *PODS.*

2006 *Inverting schema mappings.*
R. Fagin. *PODS.*

2007 *Quasi-inverses of schema mappings*
R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. *PODS.*

2008 *The recovery of a schema mapping: bringing exchanged . . .*
M. Arenas, J. Pérez, and C. Riveros. *PODS.*

2009 *Inverting schema mappings: bridging the gap . . .*
M. Arenas, J. Pérez, J. Reutter, and C. Riveros. *VLDB.*

A new semantics on how to invert a schema mapping

A new semantics on how to invert a schema mapping

Maximum recovery

- A notion of inverse based on recovering sound information.

A new semantics on how to invert a schema mapping

Maximum recovery

- A notion of inverse based on recovering sound information.
- An algorithm to compute maximum recoveries.

A new semantics on how to invert a schema mapping

Maximum recovery

- A notion of inverse based on recovering sound information.
- An algorithm to compute maximum recoveries.

$\mathcal{C}$-maximum recovery for a class of queries $\mathcal{C}$

A new semantics on how to invert a schema mapping

Maximum recovery

- A notion of inverse based on recovering sound information.
- An algorithm to compute maximum recoveries.

$\mathcal{C}$-maximum recovery for a class of queries $\mathcal{C}$

- A parameterized notion of maximum recovery.

A new semantics on how to invert a schema mapping

Maximum recovery

- A notion of inverse based on recovering sound information.
- An algorithm to compute maximum recoveries.

$\mathcal{C}$-maximum recovery for a class of queries $\mathcal{C}$

- A parameterized notion of maximum recovery.
- An algorithm to compute **CQ**-maximum recoveries.

Outline

Outline

Schema mappings are the building blocks to perform data exchange

Schema mappings are the building blocks to perform data exchange

S **T**

Two database schemas: **S** (*source*) and **T** (*target*).

Schema mappings are the building blocks to perform data exchange

Two database schemas: **S** (*source*) and **T** (*target*).

Schema mappings are the building blocks to perform data exchange

Two database schemas: **S** (*source*) and **T** (*target*).

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

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

Schema mappings are the building blocks to perform data exchange

Two database schemas: **S** (*source*) and **T** (*target*).

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

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

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

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

Schema mappings are usually given in the form of logical specifications

Schema mappings are usually given in the form of logical specifications

- Source-to-target tuple-generating dependencies (st-tgds):

$$\varphi_{\mathbf{S}}(\bar{x}) \longrightarrow \psi_{\mathbf{T}}(\bar{x})$$

with $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x})$ a **CQ**-query.

Schema mappings are usually given in the form of logical specifications

- Source-to-target tuple-generating dependencies (st-tgds):

$$\varphi_{\mathbf{S}}(\bar{x}) \longrightarrow \psi_{\mathbf{T}}(\bar{x})$$

with $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x})$ a **CQ**-query.

Example

Source: $\{$ClientA(**name**, **balance**, **city**)$\}$

Target: $\{$ClientB(**id**, **name**, **balance**, **account**)$\}$

Schema mappings are usually given in the form of logical specifications

- Source-to-target tuple-generating dependencies (st-tgds):

$$\varphi_{\mathbf{S}}(\bar{x}) \longrightarrow \psi_{\mathbf{T}}(\bar{x})$$

with $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x})$ a **CQ**-query.

Example

Source: $\{$ClientA(**name**, **balance**, **city**)$\}$

Target: $\{$ClientB(**id**, **name**, **balance**, **account**)$\}$

$$\exists Z\ \text{ClientA}(x, y, Z) \longrightarrow \exists U \exists V\ \text{ClientB}(U, x, y, V)$$

Schema mappings are usually given in the form of logical specifications

- Source-to-target tuple-generating dependencies (st-tgds):

$$\varphi_{\mathbf{S}}(\bar{x}) \longrightarrow \psi_{\mathbf{T}}(\bar{x})$$

with $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x})$ a **CQ**-query.

Example

Source: $\{$ClientA(**name**, **balance**, **city**)$\}$

Target: $\{$ClientB(**id**, **name**, **balance**, **account**)$\}$

$$\text{ClientA}(x, y, Z) \longrightarrow \exists U \exists V \ \text{ClientB}(U, x, y, V)$$

Schema mappings are usually given in the form of logical specifications

- Source-to-target tuple-generating dependencies (st-tgds):

$$\varphi_{\mathbf{S}}(\bar{x}) \longrightarrow \psi_{\mathbf{T}}(\bar{x})$$

with $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x})$ a **CQ**-query.

Example

Source: $\{$ClientA(**name**, **balance**, **city**)$\}$

Target: $\{$ClientB(**id**, **name**, **balance**, **account**)$\}$

$$\text{ClientA}(x, y, Z) \longrightarrow \exists U \exists V\ \text{ClientB}(U, x, y, V)$$

- A set Σ of dependencies *specifies* $\mathcal{M}$:

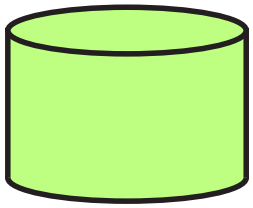
$$(I, J) \in \mathcal{M} \iff (I, J) \models \Sigma.$$

Our goal is to find the semantics

of the notion of inverse

Our goal is to find the semantics

of the notion of inverse

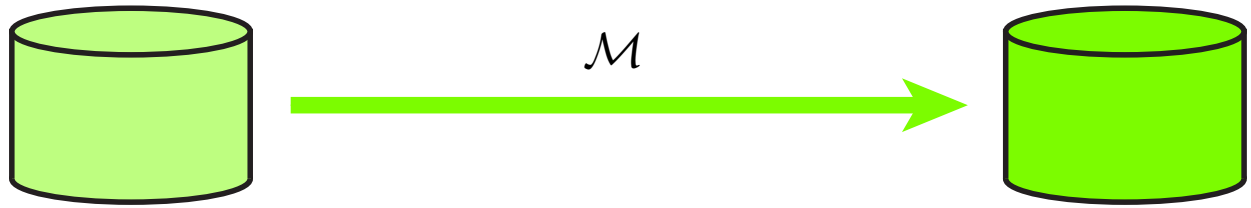


Our goal is to find the semantics

of the notion of inverse

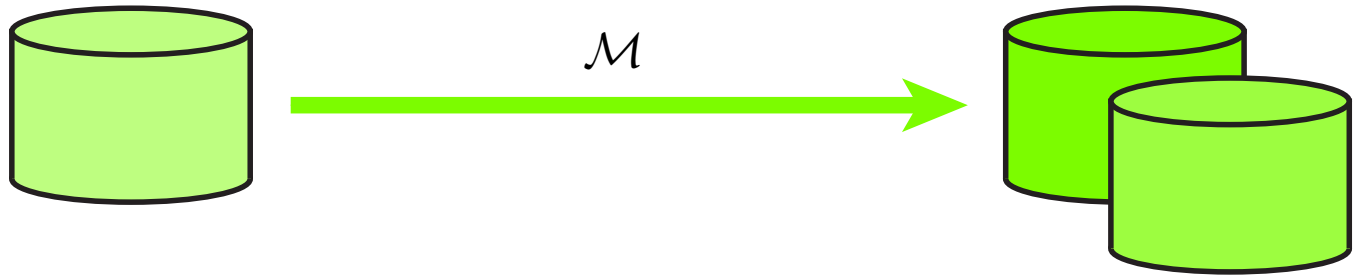
Our goal is to find the semantics

of the notion of inverse



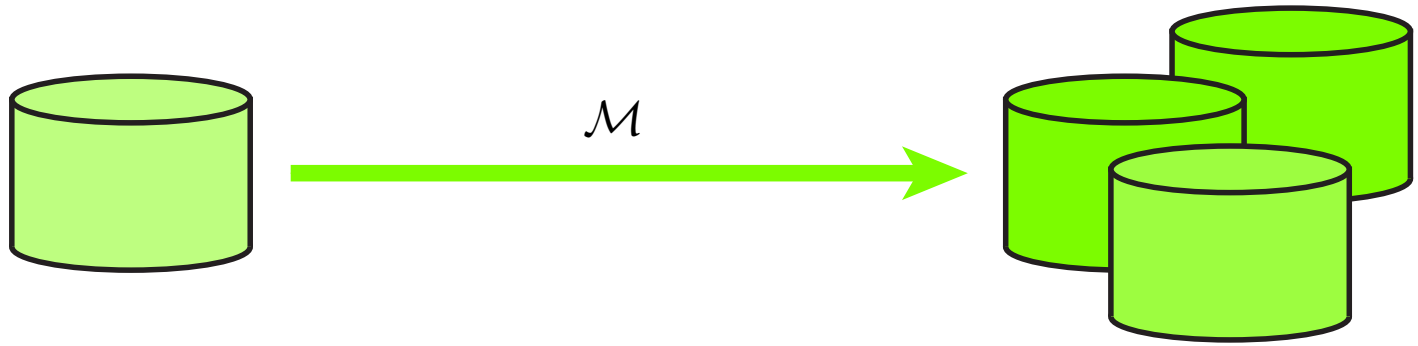
Our goal is to find the semantics

of the notion of inverse



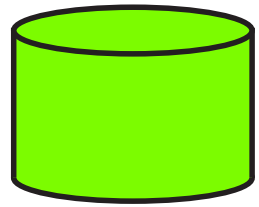
Our goal is to find the semantics

of the notion of inverse



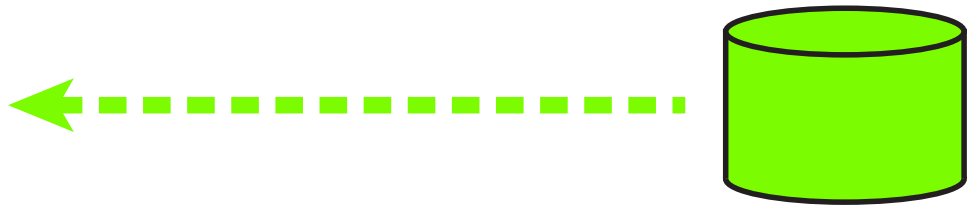
Our goal is to find the semantics

of the notion of inverse



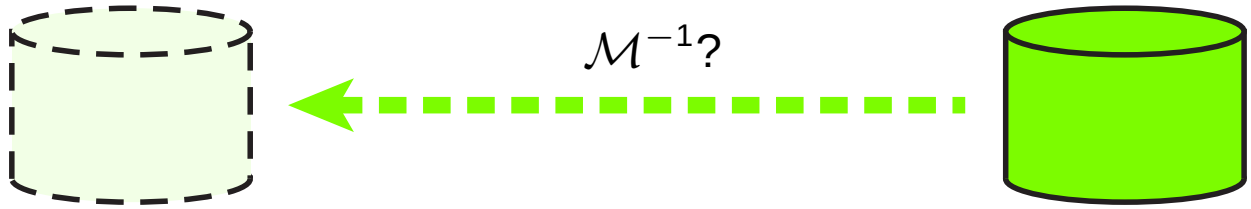
Our goal is to find the semantics

of the notion of inverse

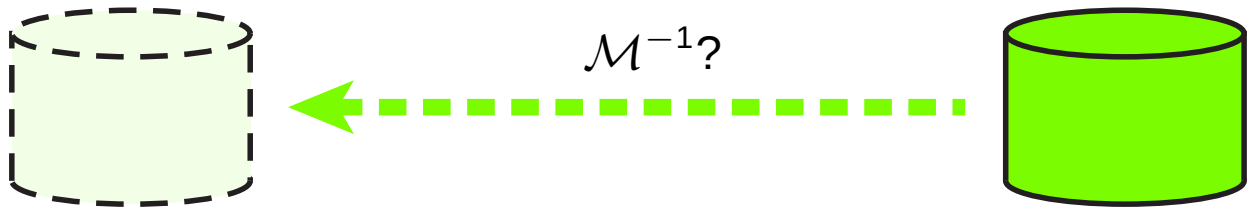


Our goal is to find the semantics

of the notion of inverse



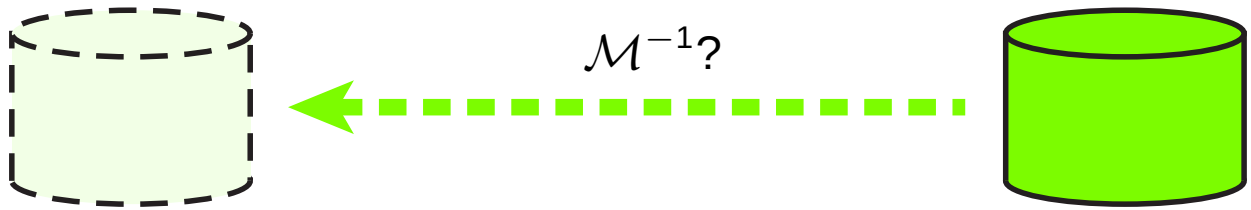
Our goal is to find the semantics of the notion of inverse



Example

$\mathcal{M}$: $\quad A(x,y) \rightarrow \exists Z\ B(x,Z)$

Our goal is to find the semantics of the notion of inverse

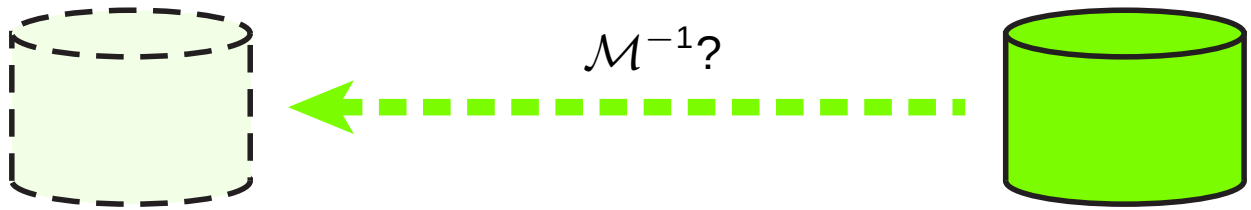


Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

I: $\{ A(1,1) \}$

Our goal is to find the semantics of the notion of inverse



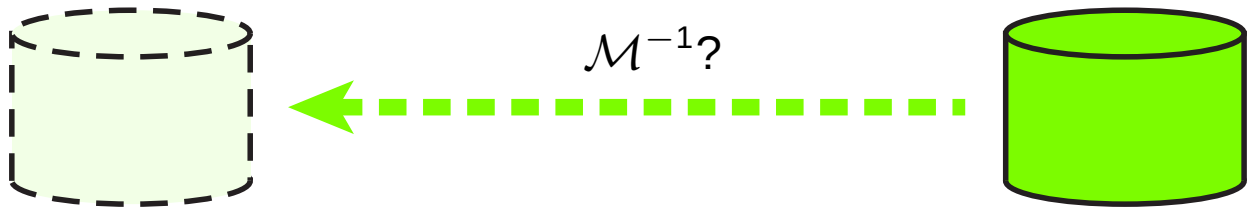
Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

I: $\{ A(1,1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{ B(1,2) \}$

Our goal is to find the semantics of the notion of inverse



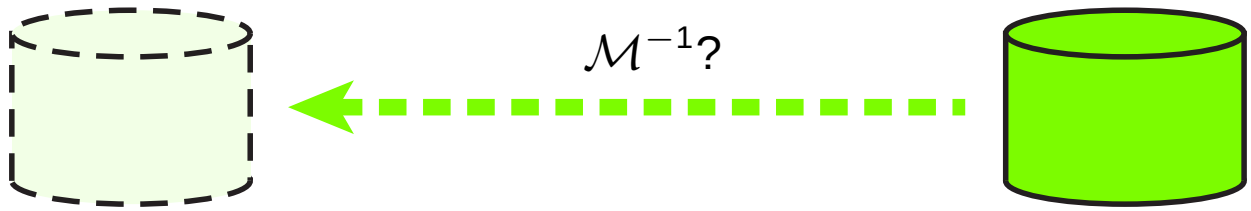
Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

I: $\{ A(1,1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{ B(1,2) \}, \{ B(1,1), B(2,3) \}$

Our goal is to find the semantics of the notion of inverse



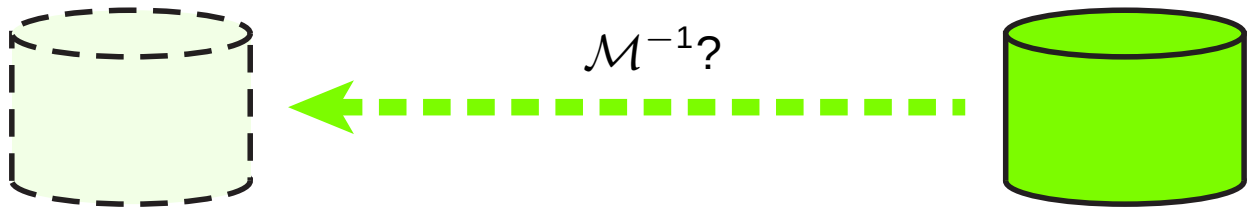
Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

I: $\{ A(1,1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{ B(1,2) \}, \{ B(1,1), B(2,3) \}, \ldots$

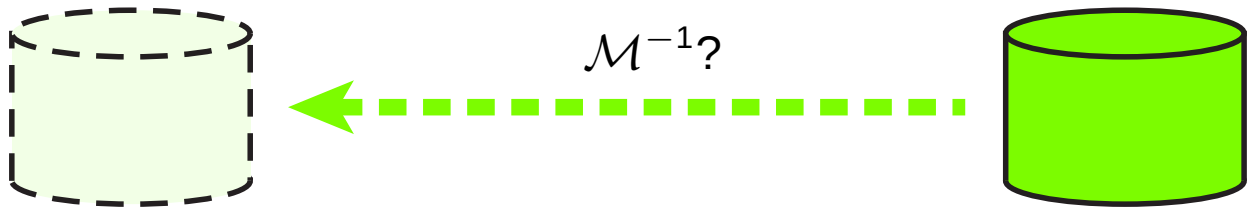
Our goal is to find the semantics of the notion of inverse



Example

$\mathcal{M}$:	$A(x,y) \rightarrow \exists Z\ B(x,Z)$
I:	$\{ A(1,1) \}$
$\mathrm{Sol}_{\mathcal{M}}(I)$:	$\{ B(1,2) \}, \{ B(1,1), B(2,3) \}, \ldots$
J:	$\{ B(1,2) \}$

Our goal is to find the semantics of the notion of inverse



Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

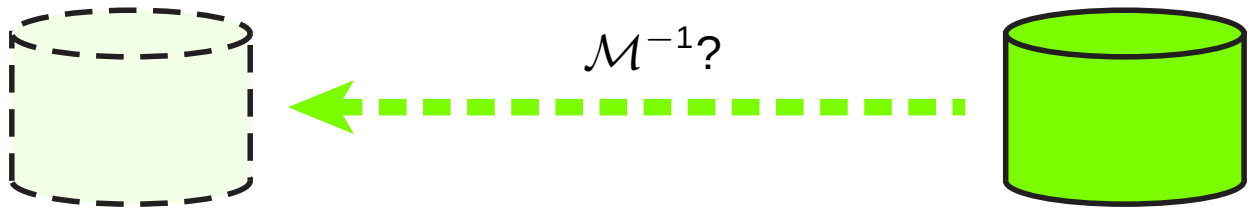
I: $\{ A(1,1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{ B(1,2) \}, \{ B(1,1), B(2,3) \}, \ldots$

J: $\{ B(1,2) \}$

$\mathrm{Sol}_{\mathcal{M}^{-1}}(J)$: $\{ A(1,1) \}$

Our goal is to find the semantics of the notion of inverse



Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

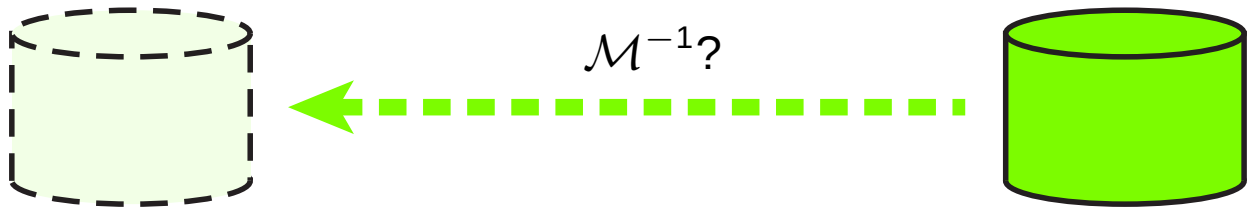
I: $\{ A(1,1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{ B(1,2) \}, \{ B(1,1), B(2,3) \}, \ldots$

J: $\{ B(1,2) \}$

$\mathrm{Sol}_{\mathcal{M}^{-1}}(J)$: $\{ A(1,1) \}, \{ A(1,2), A(1,3) \}$

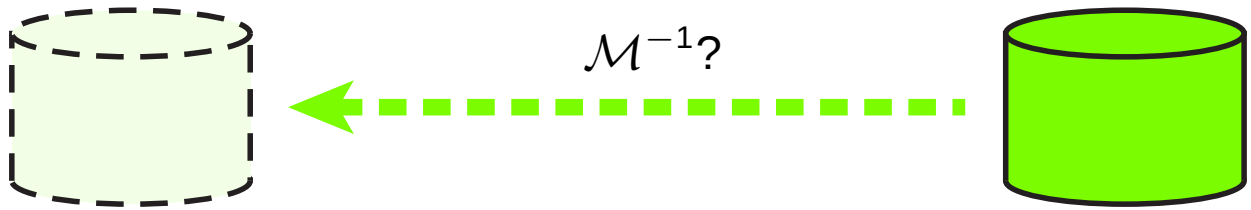
Our goal is to find the semantics of the notion of inverse



Example

$\mathcal{M}$:	$A(x,y) \rightarrow \exists Z\ B(x,Z)$
I:	$\{ A(1,1) \}$
$\mathrm{Sol}_{\mathcal{M}}(I)$:	$\{ B(1,2) \}, \{ B(1,1), B(2,3) \}, \ldots$
J:	$\{ B(1,2) \}$
$\mathrm{Sol}_{\mathcal{M}^{-1}}(J)$:	$\{ A(1,1) \}, \{ A(1,2), A(1,3) \}, \{ \emptyset \}, \ldots$

Our goal is to find the semantics of the notion of inverse



Example

$\mathcal{M}$: $A(x,y) \rightarrow \exists Z\ B(x,Z)$

I: $\{ A(1,1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{ B(1,2) \}, \{ B(1,1), B(2,3) \}, \ldots$

J: $\{ B(1,2) \}$

$\mathrm{Sol}_{\mathcal{M}^{-1}}(J)$: $\{ A(1,1) \}, \{ A(1,2), A(1,3) \}, \{ \emptyset \}, \ldots$

What is the mapping from B to A ?

We want a consistent and useful semantics of the notion of inverse

We want a consistent and useful semantics of the notion of inverse

We want a semantics of the notion of inverse, such that:

We want a consistent and useful semantics of the notion of inverse

We want a semantics of the notion of inverse, such that:

- we can recover the exchanged data (or part of it).

We want a consistent and useful semantics of the notion of inverse

We want a semantics of the notion of inverse, such that:

- we can recover the exchanged data (or part of it).
- we can specify it in the same language (or slightly different).

We want a consistent and useful semantics of the notion of inverse

We want a semantics of the notion of inverse, such that:

- we can recover the exchanged data (or part of it).
- we can specify it in the same language (or slightly different).
- we can always compute it.

How can we give a consistent semantics to the notion of inverse?

We want to define

what means to recover sound information

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$\mathcal{M}$: $\quad \mathsf{A}(x, y, Z, Z) \quad \longrightarrow \quad \exists U \;\; \mathsf{B}\,(U, x, y)$

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$\mathcal{M}$: $\quad \mathsf{A}(x, y, Z, Z) \quad \longrightarrow \quad \exists U \;\; \mathsf{B}\,(U, x, y)$

$\mathcal{M}_1$: $\quad \mathsf{B}(U, x, y) \quad \longrightarrow \quad \exists Y_1 \; \exists Z_1 \; \exists Z_2 \quad \mathsf{A}(x, Y_1, Z_1, Z_2)$

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$\mathcal{M}$: $\quad \mathsf{A}(x, y, Z, Z) \longrightarrow \exists U \ \mathsf{B}(U, x, y)$

$\mathcal{M}_1$: $\quad \mathsf{B}(U, x, y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathsf{A}(x, Y_1, Z_1, Z_2) \quad \checkmark$

We want to define
what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$$\mathcal{M}: \quad \mathsf{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathsf{B}(U,x,y)$$

$$\mathcal{M}_1: \quad \mathsf{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathsf{A}(x,Y_1,Z_1,Z_2) \quad \checkmark$$

$$\mathcal{M}_2: \quad \mathsf{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 \quad \mathsf{A}(x,y,Z_1,Z_2)$$

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$$\mathcal{M}: \quad \mathsf{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathsf{B}(U,x,y)$$

$$\mathcal{M}_1: \quad \mathsf{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathsf{A}(x,Y_1,Z_1,Z_2) \quad \checkmark$$

$$\mathcal{M}_2: \quad \mathsf{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 \quad \mathsf{A}(x,y,Z_1,Z_2) \quad \checkmark$$

We want to define
what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$$\mathcal{M}: \quad \mathsf{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathsf{B}(U,x,y)$$

$$\begin{array}{lllll} \mathcal{M}_1: & \mathsf{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 & \mathsf{A}(x,Y_1,Z_1,Z_2) & \checkmark \\ \mathcal{M}_2: & \mathsf{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 & \mathsf{A}(x,y,Z_1,Z_2) & \checkmark \\ \mathcal{M}_3: & \mathsf{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 & \mathsf{A}(x,Z_1,y,Z_2) & \end{array}$$

We want to define

what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\begin{array}{lllll}
\mathcal{M}_1: & \mathrm{B}(U,x,y) \longrightarrow & \exists Y_1 \, \exists Z_1 \, \exists Z_2 & \mathrm{A}(x,Y_1,Z_1,Z_2) & \checkmark \\
\mathcal{M}_2: & \mathrm{B}(U,x,y) \longrightarrow & \exists Z_1 \, \exists Z_2 & \mathrm{A}(x,y,Z_1,Z_2) & \checkmark \\
\mathcal{M}_3: & \mathrm{B}(U,x,y) \longrightarrow & \exists Z_1 \, \exists Z_2 & \mathrm{A}(x,Z_1,y,Z_2) & \times
\end{array}$$

We want to define
what means to recover sound information

- data *may be lost* in the exchange through $\mathcal{M}$.
- we want an $\mathcal{M}'$ that *at least* recovers sound data w.r.t. $\mathcal{M}$.

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\begin{array}{lllll}
\mathcal{M}_1: & \mathrm{B}(U,x,y) \longrightarrow & \exists Y_1 \exists Z_1 \exists Z_2 & \mathrm{A}(x,Y_1,Z_1,Z_2) & \checkmark \\
\mathcal{M}_2: & \mathrm{B}(U,x,y) \longrightarrow & \exists Z_1 \exists Z_2 & \mathrm{A}(x,y,Z_1,Z_2) & \checkmark \\
\mathcal{M}_3: & \mathrm{B}(U,x,y) \longrightarrow & \exists Z_1 \exists Z_2 & \mathrm{A}(x,Z_1,y,Z_2) & \times
\end{array}$$

We call mapping $\mathcal{M}_1$ and $\mathcal{M}_2$ *recoveries* of $\mathcal{M}$.

A mapping is a *recovery* of $\mathcal{M}$ if

it recovers sound information wrt to $\mathcal{M}$

A mapping is a *recovery* of $\mathcal{M}$ if it recovers sound information wrt to $\mathcal{M}$

Definition

A mapping $\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ iff for every source instance I:

$$(I, I) \in \mathcal{M} \circ \mathcal{M}'$$

A mapping is a *recovery* of $\mathcal{M}$ if it recovers sound information wrt to $\mathcal{M}$

Definition

A mapping $\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ iff for every source instance I:

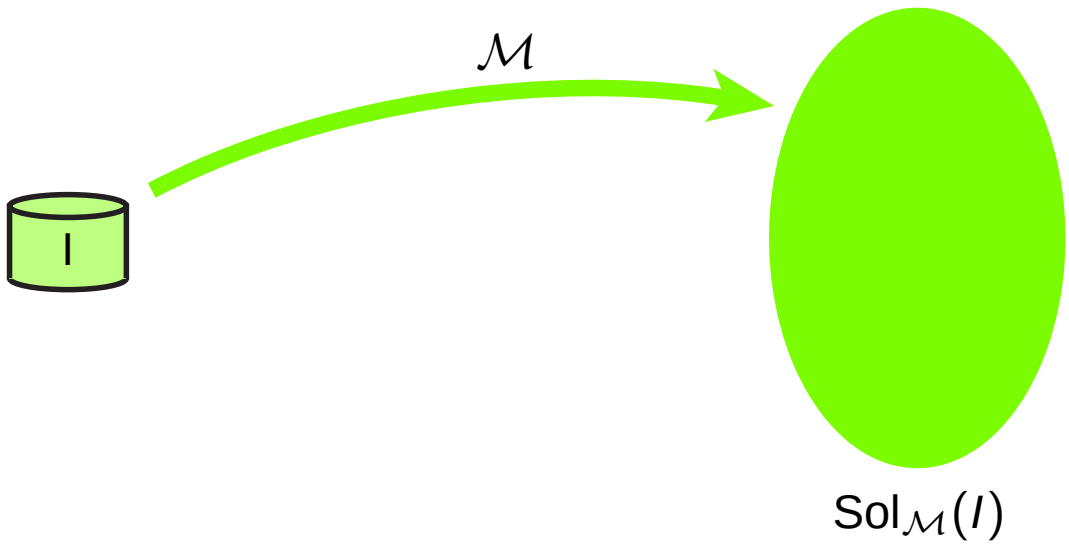
$$(I, I) \in \mathcal{M} \circ \mathcal{M}'$$

A mapping is a *recovery* of $\mathcal{M}$ if it recovers sound information wrt to $\mathcal{M}$

Definition

A mapping $\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ iff for every source instance I:

$$(I, I) \in \mathcal{M} \circ \mathcal{M}'$$

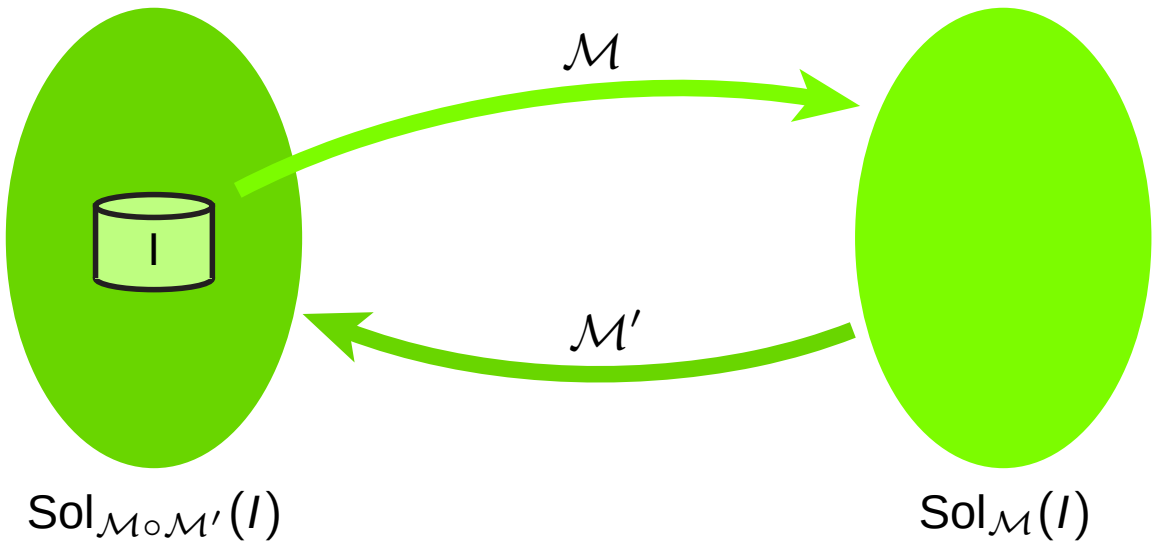


A mapping is a *recovery* of $\mathcal{M}$ if it recovers sound information wrt to $\mathcal{M}$

Definition

A mapping $\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ iff for every source instance I:

$$(I, I) \in \mathcal{M} \circ \mathcal{M}'$$



We want to choose

the most informative recovery of $\mathcal{M}$

We want to choose
the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

We want to choose
the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

Example

$\mathcal{M}$: $\quad \mathrm{A}(x, y, Z, Z) \quad \longrightarrow \quad \exists U \ \mathrm{B}(U, x, y)$

We want to choose the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\mathcal{M}_1: \quad \mathrm{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,Y_1,Z_1,Z_2)$$

$$\mathcal{M}_2: \quad \mathrm{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,y,Z_1,Z_2)$$

We want to choose the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\mathcal{M}_1: \quad \mathrm{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,Y_1,Z_1,Z_2)$$

$$\mathcal{M}_2: \quad \mathrm{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,y,Z_1,Z_2)$$

$\mathcal{M}_2$ is better than $\mathcal{M}_1$

We want to choose the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\begin{aligned}
\mathcal{M}_1: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,Y_1,Z_1,Z_2) \\
\mathcal{M}_2: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,y,Z_1,Z_2) \\
\mathcal{M}_4: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Z \quad \mathrm{A}(x,y,Z,Z)
\end{aligned}$$

$\mathcal{M}_2$ is better than $\mathcal{M}_1$

We want to choose the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\begin{aligned}
\mathcal{M}_1: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Y_1 \, \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,Y_1,Z_1,Z_2) \\
\mathcal{M}_2: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Z_1 \, \exists Z_2 \quad \mathrm{A}(x,y,Z_1,Z_2) \\
\mathcal{M}_4: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Z \quad \mathrm{A}(x,y,Z,Z)
\end{aligned}$$

$\mathcal{M}_2$ *is better than* $\mathcal{M}_1$

$\mathcal{M}_4$ *is better than* $\mathcal{M}_2$ and $\mathcal{M}_1$

We want to choose the most informative recovery of $\mathcal{M}$

- Can we compare alternatives recoveries?

Example

$$\mathcal{M}: \quad \mathrm{A}(x,y,Z,Z) \longrightarrow \exists U \ \mathrm{B}(U,x,y)$$

$$\begin{aligned}
\mathcal{M}_1: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Y_1 \exists Z_1 \exists Z_2 \quad \mathrm{A}(x,Y_1,Z_1,Z_2)\\
\mathcal{M}_2: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Z_1 \exists Z_2 \quad \mathrm{A}(x,y,Z_1,Z_2)\\
\mathcal{M}_4: &\quad \mathrm{B}(U,x,y) \longrightarrow \exists Z \quad \mathrm{A}(x,y,Z,Z)
\end{aligned}$$

$\mathcal{M}_2$ *is better than* $\mathcal{M}_1$

$\mathcal{M}_4$ *is better than* $\mathcal{M}_2$ and $\mathcal{M}_1$

We call mapping $\mathcal{M}_4$ a *maximum recovery* of $\mathcal{M}$.

A mapping is a *maximum recovery* of $\mathcal{M}$

if it is the *smallest* recovery of $\mathcal{M}$

A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

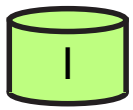
$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$

A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$

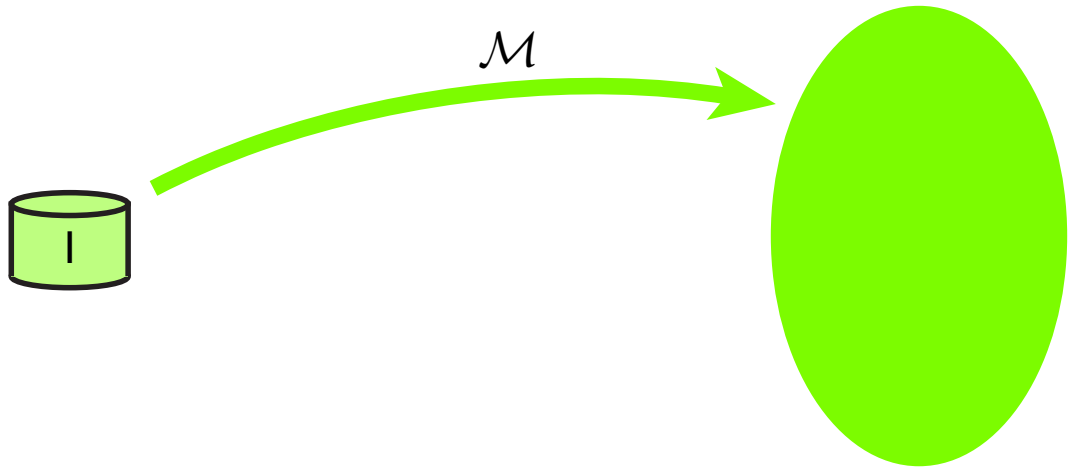


A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$

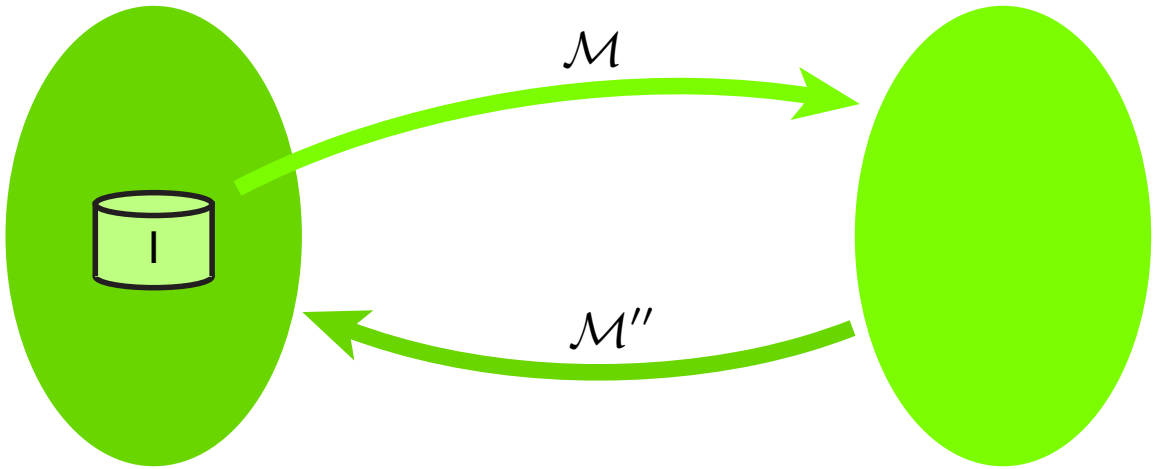


A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$

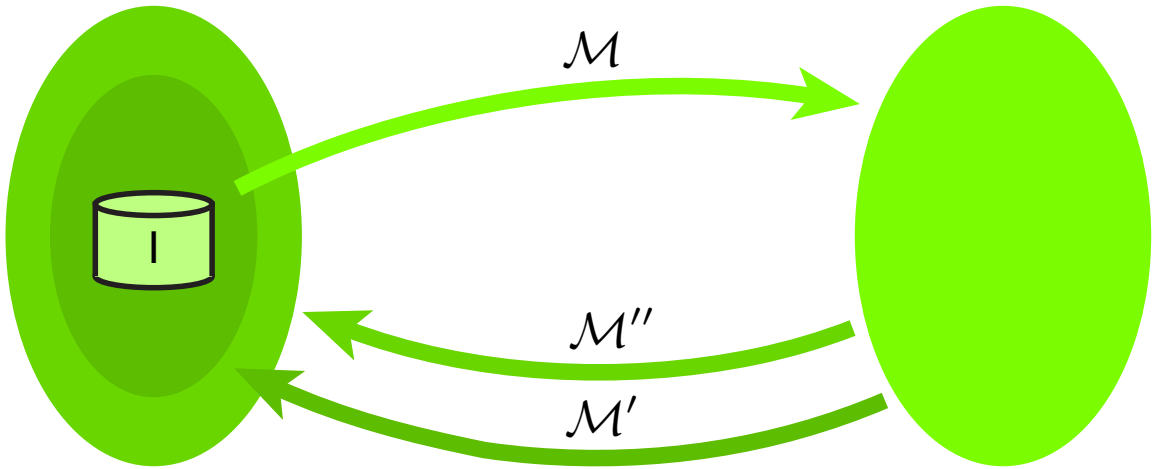


A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$

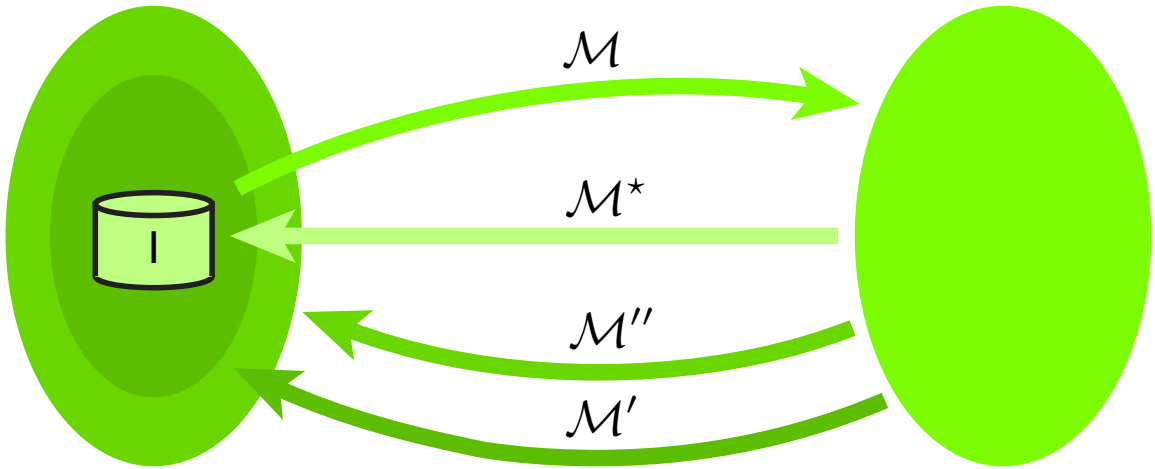


A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$

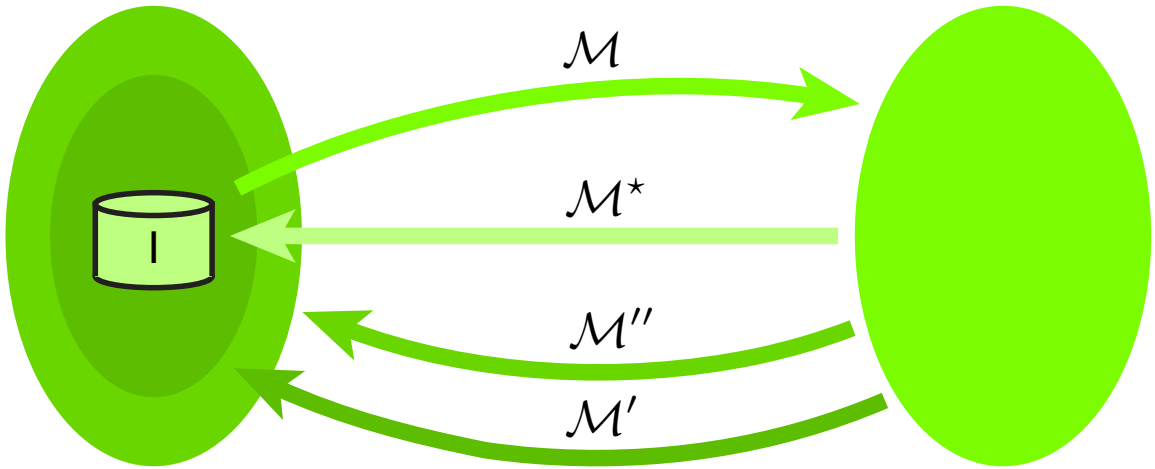


A mapping is a *maximum recovery* of $\mathcal{M}$ if it is the *smallest* recovery of $\mathcal{M}$

Definition

Given a $\mathcal{M}'$ and $\mathcal{M}''$ recoveries of $\mathcal{M}$, we say that $\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$ or

$$\mathcal{M}'' \preceq \mathcal{M}' \quad \text{if and only if} \quad \mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''.$$



Definition

$\mathcal{M}'$ is a *maximum recovery* of $\mathcal{M}$ if $\mathcal{M}'$ is a maximum w.r.t. $\preceq$.

Maximum recovery strictly generalize previous notions

Maximum recovery strictly generalize previous notions

In [Fag06], Fagin gives the first definition of an inverse of a mapping:

- Focused on the relational case and st-tgds.

Maximum recovery strictly generalize previous notions

In [Fag06], Fagin gives the first definition of an inverse of a mapping:

- Focused on the relational case and st-tgds.
- Limitation: too restrictive.

Maximum recovery strictly generalize previous notions

In [Fag06], Fagin gives the first definition of an inverse of a mapping:

- Focused on the relational case and st-tgds.
- Limitation: too restrictive.

Theorem

Let $\mathcal{M}$ be a mappings specified by st-tgds and fagin-invertible, then:

$\mathcal{M}'$ is an fagin-inverse of $\mathcal{M} \iff \mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.

Maximum recovery strictly generalize previous notions

In [Fag06], Fagin gives the first definition of an inverse of a mapping:

- Focused on the relational case and st-tgds.
- Limitation: too restrictive.

Theorem

Let $\mathcal{M}$ be a mappings specified by st-tgds and fagin-invertible, then:

$\mathcal{M}'$ is an fagin-inverse of $\mathcal{M} \iff \mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.

Advantage of maximum recoveries:

- Fagin inverses rarely exist for st-tgds.
- Maximum recoveries always exist for st-tgds.

How can we compute maximum recoveries?

Certain answers and query rewriting are the key concepts in the algorithm

Certain answers and query rewriting are the key concepts in the algorithm

Certain answers and *query rewriting*:

- are old concepts in data integration.

Certain answers and query rewriting are the key concepts in the algorithm

Certain answers and *query rewriting*:

- are old concepts in data integration.
- are well studied for conjunctive queries.

Certain answers and query rewriting are the key concepts in the algorithm

Certain answers and *query rewriting*:

- are old concepts in data integration.
- are well studied for conjunctive queries.
- are the ingredients needed to compute maximum recoveries.

Certain answers: tuple present in all solutions

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$$\mathcal{M}: \qquad A(x, y) \rightarrow \exists Z \; B(x, Z)$$

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$\mathcal{M}$: $\quad A(x, y) \rightarrow \exists Z\ B(x, Z)$

I: $\quad \{ A(1, 1) \}$

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$\mathcal{M}$: $A(x, y) \rightarrow \exists Z\ B(x, Z)$

I: $\{ A(1, 1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{B(1, 1)\}, \{B(1, 2)\}, \{B(1, \perp)\}, \ldots$

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$\mathcal{M}$: $A(x, y) \rightarrow \exists Z\ B(x, Z)$

I: $\{ A(1, 1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{B(1, 1)\}, \{B(1, 2)\}, \{B(1, \perp)\}, \ldots$

$Q_1(u, v)$: $B(u, v)$

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$\mathcal{M}$: $A(x, y) \rightarrow \exists Z\ B(x, Z)$

I: $\{ A(1, 1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{B(1, 1)\}, \{B(1, 2)\}, \{B(1, \perp)\}, \ldots$

$Q_1(u, v)$: $B(u, v)$

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

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$\mathcal{M}$: $A(x, y) \rightarrow \exists Z\ B(x, Z)$

I: $\{ A(1, 1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{B(1, 1)\}, \{B(1, 2)\}, \{B(1, \perp)\}, \ldots$

$Q_2(u)$: $\exists Z\ B(u, Z)$

Certain answers: tuple present in all solutions

Definition

Given a mapping $\mathcal{M}$ and a source instance I, we say that $\bar{t}$ is a *certain answer* for Q over I iff

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

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

Example

$\mathcal{M}$: $A(x, y) \rightarrow \exists Z\ B(x, Z)$

I: $\{ A(1, 1) \}$

$\mathrm{Sol}_{\mathcal{M}}(I)$: $\{B(1, 1)\}, \{B(1, 2)\}, \{B(1, \perp)\}, \ldots$

$Q_2(u)$: $\exists Z\ B(u, Z)$

$\underline{\text{certain}}_{\mathcal{M}}(Q_2, I) = \{1\}$

Query rewriting:

reformulating a target query in the source

Query rewriting:
reformulating a target query in the source

Definition

Given a mapping $\mathcal{M}$ and a target query Q, we say that Q' is a *source rewriting* of Q under $\mathcal{M}$ if

Query rewriting:
reformulating a target query in the source

Definition

Given a mapping $\mathcal{M}$ and a target query Q, we say that Q' is a *source rewriting* of Q under $\mathcal{M}$ if

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

Query rewriting:
reformulating a target query in the source

Definition

Given a mapping $\mathcal{M}$ and a target query Q, we say that Q' is a *source rewriting* of Q under $\mathcal{M}$ if

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

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

Computing a maximum recovery using rewriting of queries

Computing a maximum recovery using rewriting of queries

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

Computing a maximum recovery using rewriting of queries

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

Algorithm

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

Computing a maximum recovery using rewriting of queries

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

Algorithm

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 source rewriting of $\exists \bar{y} \psi(\bar{x}, \bar{y})$ under $\mathcal{M}$.

Computing a maximum recovery using rewriting of queries

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

Algorithm

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 source rewriting of $\exists \bar{y} \psi(\bar{x}, \bar{y})$ under $\mathcal{M}$.
- Add to the definition of $\mathcal{M}'$ the dependency

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

Computing a maximum recovery using rewriting of queries

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

Algorithm

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 source rewriting of $\exists \bar{y} \psi(\bar{x}, \bar{y})$ under $\mathcal{M}$.
- Add to the definition of $\mathcal{M}'$ the dependency

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

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

Maximum recovery algorithm

is quadratic in the full case, exponential in general

Maximum recovery algorithm

is quadratic in the full case, exponential in general

	full st-tgds	st-tgds
Output:	**CQ-to-UCQ**$^{=}$	**CQ**$^{\mathbf{C}(\cdot)}$**-to-UCQ**$^{=}$
Size:	quadratic	exponential
Time:	quadratic	exponential

Maximum recovery algorithm

is quadratic in the full case, exponential in general

	full st-tgds	st-tgds
Output:	**CQ-to-UCQ**$^{=}$	**CQ**$^{\mathbf{C}(\cdot)}$**-to-UCQ**$^{=}$
Size:	quadratic	exponential
Time:	quadratic	exponential

Some highlights:

- We use query-rewriting as a black box in the algorithm.

Maximum recovery algorithm

is quadratic in the full case, exponential in general

	full st-tgds	st-tgds
Output:	**CQ-to-UCQ**$^{=}$	**CQ**$^{\mathbf{C}(\cdot)}$**-to-UCQ**$^{=}$
Size:	quadratic	exponential
Time:	quadratic	exponential

Some highlights:

- We use query-rewriting as a black box in the algorithm.
- Disjunction and predicate $\mathbf{C}(\cdot)$ is the language needed to specify maximum recoveries for st-tgds.

We need to extend the language of st-tgds in order to express maximum recoveries

We need to extend the language of st-tgds in order to express maximum recoveries

Problems:

- The algorithm returns a set of **CQ**$^{\mathbf{C}(\cdot)}$**-to-UCQ**$^{=}$ dependencies.

We need to extend the language of st-tgds in order to express maximum recoveries

Problems:

- The algorithm returns a set of $\mathbf{CQ}^{\mathbf{C}(\cdot)}$**-to-**$\mathbf{UCQ}^{=}$ dependencies.
- Disjunctions are unavoidable to express maximum recoveries.

We need to extend the language of st-tgds in order to express maximum recoveries

Problems:

- The algorithm returns a set of $\mathbf{CQ}^{\mathbf{C}(\cdot)}$**-to-**$\mathbf{UCQ}^{=}$ dependencies.
- Disjunctions are unavoidable to express maximum recoveries.
- How do we exchange data using st-tgds with disjunctions?

We need to extend the language of st-tgds in order to express maximum recoveries

Problems:

- The algorithm returns a set of $\mathbf{CQ}^{\mathbf{C}(\cdot)}$**-to-**$\mathbf{UCQ}^{=}$ dependencies.
- Disjunctions are unavoidable to express maximum recoveries.
- How do we exchange data using st-tgds with disjunctions?

We would like a natural notion of inverse such that:

- st-tgds always have an inverse, and

We need to extend the language of st-tgds in order to express maximum recoveries

Problems:

- The algorithm returns a set of $\mathbf{CQ}^{\mathbf{C}(\cdot)}$**-to-**$\mathbf{UCQ}^{=}$ dependencies.
- Disjunctions are unavoidable to express maximum recoveries.
- How do we exchange data using st-tgds with disjunctions?

We would like a natural notion of inverse such that:

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

How can we compute an inverse without disjunctions?

We can parameterize the notion of maximum recovery by a class of queries

We can parameterize the notion of maximum recovery by a class of queries

Maximum recoveries:

- recover the maximum amount of sound information.

We can parameterize the notion of maximum recovery by a class of queries

Maximum recoveries:

- recover the maximum amount of sound information.
- recover disjunctive information that is not useful.

We can parameterize the notion of maximum recovery by a class of queries

Maximum recoveries:

- recover the maximum amount of sound information.
- recover disjunctive information that is not useful.

If we concentrate in conjunctive information as st-tgds, we can find more practical inverses.

We want to recover sound information

with respect to a query

We want to recover sound information with respect to a query

Example

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

We want to recover sound information with respect to a query

Example

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

We want to recover sound information with respect to a query

Example

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

$$I: \quad \{A(1,2,3)\}$$

We want to recover sound information with respect to a query

Example

$$\begin{aligned} \mathcal{M}: &\quad A(x,y,z) \longrightarrow \exists U\, B(x,y,U) \\ \mathcal{M}': &\quad B(x,y,u) \longrightarrow A(x,y,x) \\ \\ I: &\quad \{A(1,2,3)\} \\ \mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I): &\quad \{A(1,2,1)\},\ \{A(1,2,1), A(3,4,5)\},\ \ldots \end{aligned}$$

We want to recover sound information with respect to a query

Example

$$\begin{aligned}
\mathcal{M}: &\quad A(x,y,z) \longrightarrow \exists U\, B(x,y,U) \\
\mathcal{M}': &\quad B(x,y,u) \longrightarrow A(x,y,x) \\
\\
I: &\quad \{A(1,2,3)\} \\
\mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I): &\quad \{A(1,2,1)\},\ \{A(1,2,1), A(3,4,5)\},\ \ldots \\
\\
Q_1(x): &\quad \exists Y\ \exists Z\ A(x,Y,Z)
\end{aligned}$$

We want to recover sound information with respect to a query

Example

$$\begin{aligned}
\mathcal{M}: &\quad A(x,y,z) \longrightarrow \exists U\, B(x,y,U) \\
\mathcal{M}': &\quad B(x,y,u) \longrightarrow A(x,y,x) \\
\\
I: &\quad \{A(1,2,3)\} \\
\mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I): &\quad \{A(1,2,1)\},\ \{A(1,2,1), A(3,4,5)\},\ \ldots \\
\\
Q_1(x): &\quad \exists Y\ \exists Z\ A(x,Y,Z)\ \checkmark
\end{aligned}$$

We want to recover sound information with respect to a query

Example

$$\begin{aligned}
\mathcal{M}: &\quad A(x,y,z) \longrightarrow \exists U\, B(x,y,U) \\
\mathcal{M}': &\quad B(x,y,u) \longrightarrow A(x,y,x) \\
\\
I: &\quad \{A(1,2,3)\} \\
\mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I): &\quad \{A(1,2,1)\},\ \{A(1,2,1), A(3,4,5)\},\ \ldots \\
\\
Q_1(x): &\quad \exists Y\ \exists Z\ A(x,Y,Z) \quad \checkmark \\
Q_2(z): &\quad \exists X\ \exists Y\ A(X,Y,z)
\end{aligned}$$

We want to recover sound information with respect to a query

Example

$$\begin{aligned}
\mathcal{M}: &\quad A(x,y,z) \longrightarrow \exists U\, B(x,y,U)\\
\mathcal{M}': &\quad B(x,y,u) \longrightarrow A(x,y,x)\\
\\
I: &\quad \{A(1,2,3)\}\\
\mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I): &\quad \{A(1,2,1)\},\ \{A(1,2,1), A(3,4,5)\},\ \ldots\\
\\
Q_1(x): &\quad \exists Y\ \exists Z\ A(x,Y,Z)\quad \checkmark\\
Q_2(z): &\quad \exists X\ \exists Y\ A(X,Y,z)\quad \times
\end{aligned}$$

We want to recover sound information with respect to a query

Example

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

$$\begin{aligned}
I: &\quad \{A(1,2,3)\} \\
\mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I): &\quad \{A(1,2,1)\},\ \{A(1,2,1), A(3,4,5)\},\ \ldots
\end{aligned}$$

$$\begin{aligned}
Q_1(x): &\quad \exists Y\ \exists Z\ A(x,Y,Z) \quad \checkmark \\
Q_2(z): &\quad \exists X\ \exists Y\ A(X,Y,z) \quad \times
\end{aligned}$$

We call mapping $\mathcal{M}'$ a Q_1-*recovery* of $\mathcal{M}$.

A mapping is a *Q-recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

A mapping is a *Q-recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a *Q-recovery* of $\mathcal{M}$ iff for every source instance I:

A mapping is a Q-*recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a Q-*recovery* of $\mathcal{M}$ iff for every source instance I:

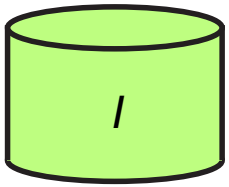
$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) \subseteq Q(I)$$

A mapping is a Q-*recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a Q-*recovery* of $\mathcal{M}$ iff for every source instance I:

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

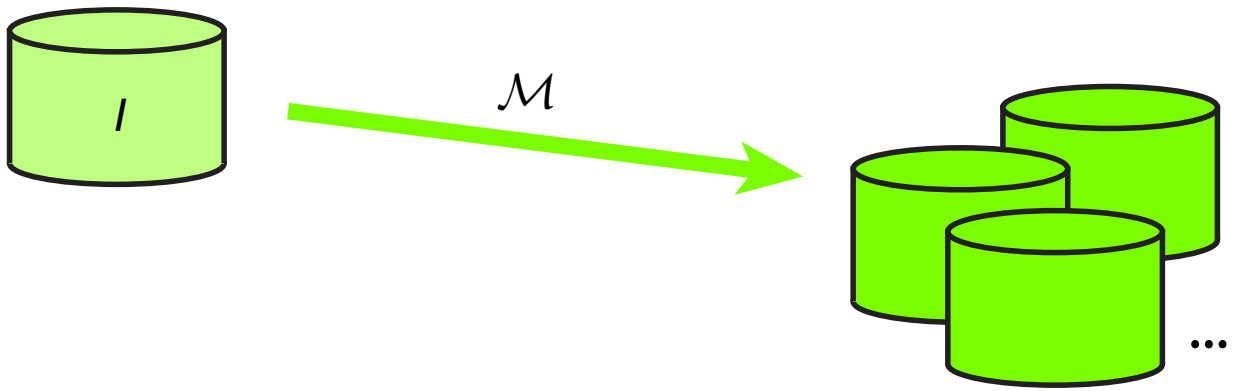


A mapping is a *Q-recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a *Q-recovery* of $\mathcal{M}$ iff for every source instance I:

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

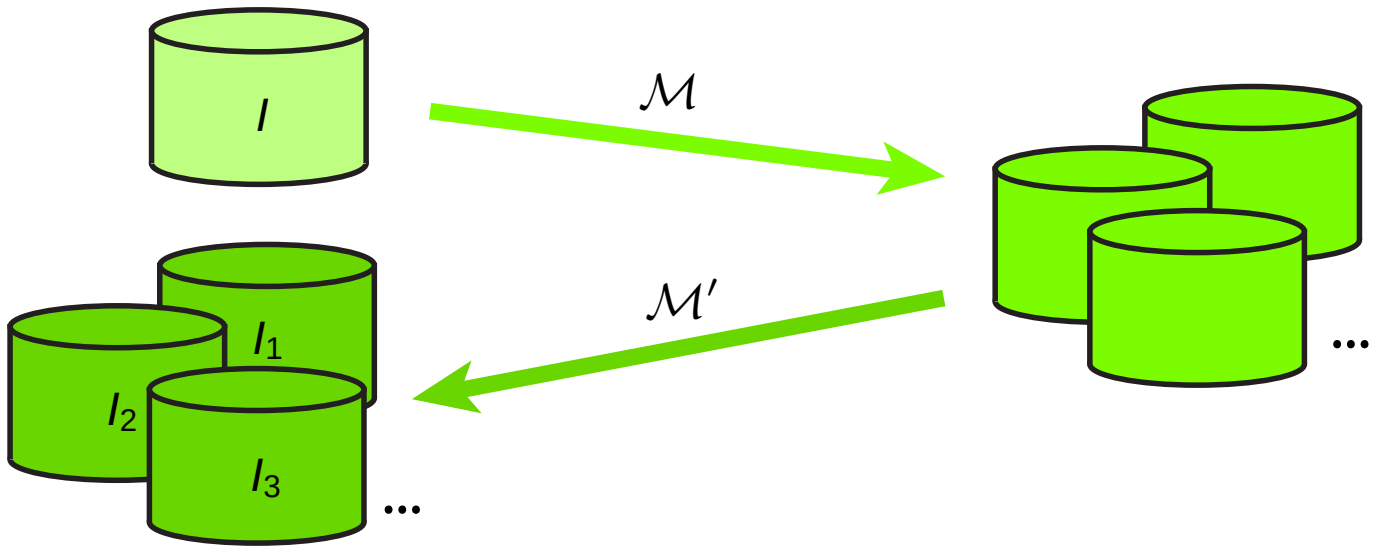


A mapping is a *Q-recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a *Q-recovery* of $\mathcal{M}$ iff for every source instance I:

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

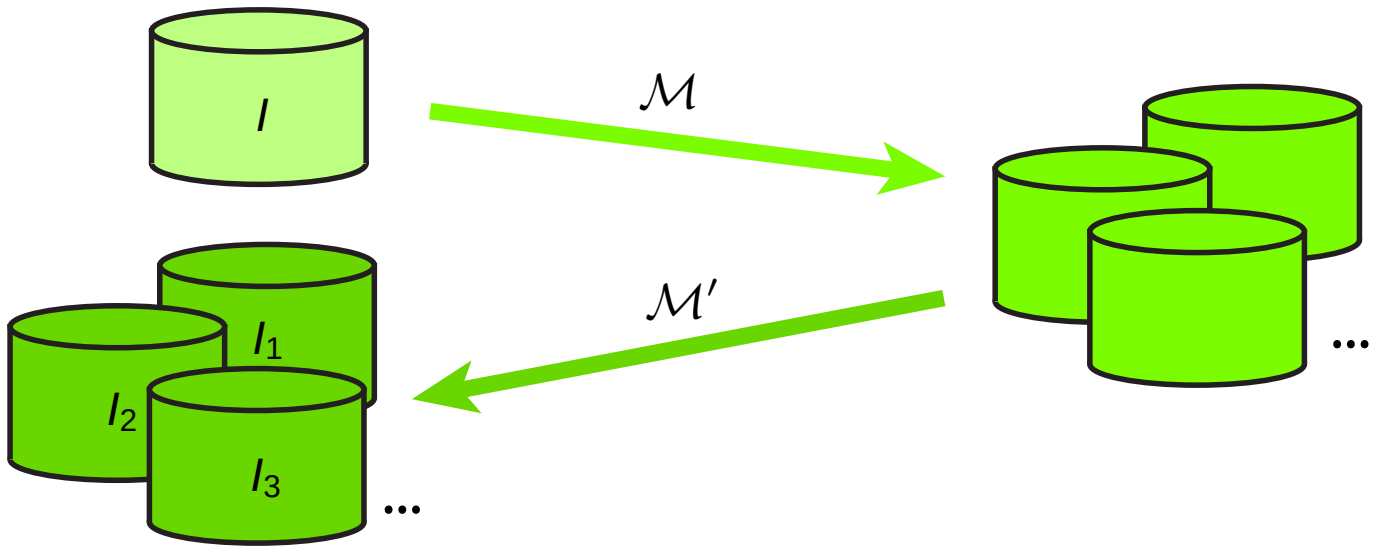


A mapping is a Q-*recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a Q-*recovery* of $\mathcal{M}$ iff for every source instance I:

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



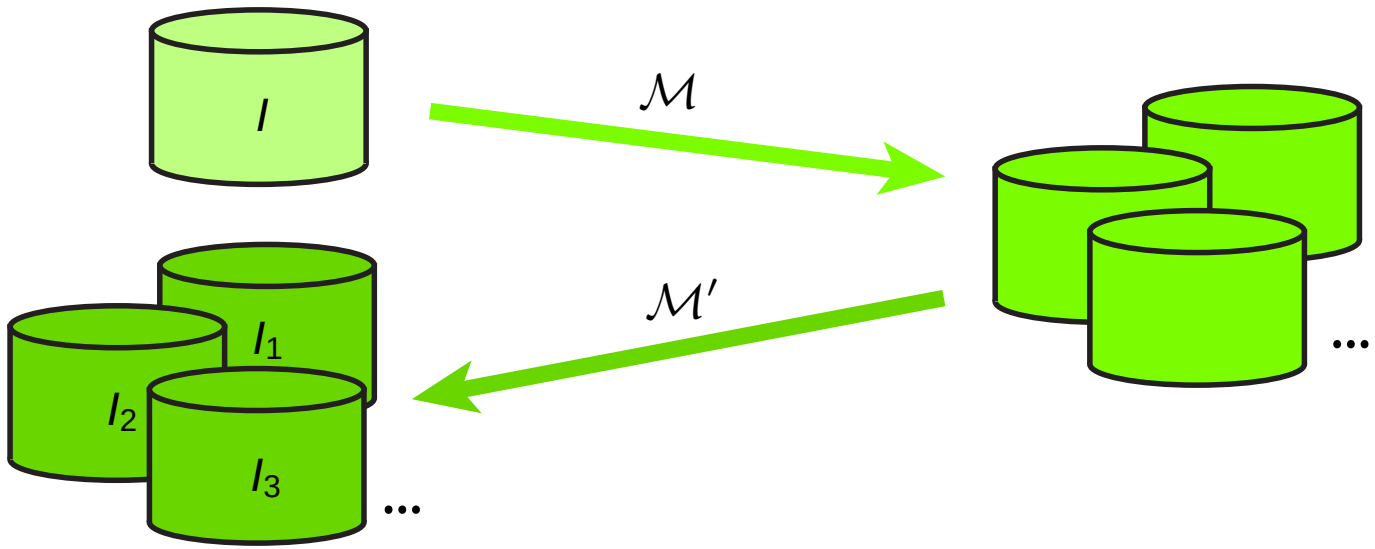
$$\underbrace{Q(I_1) \cap Q(I_2) \cap Q(I_3) \cap \cdots}$$

A mapping is a *Q-recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a *Q-recovery* of $\mathcal{M}$ iff for every source instance I:

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



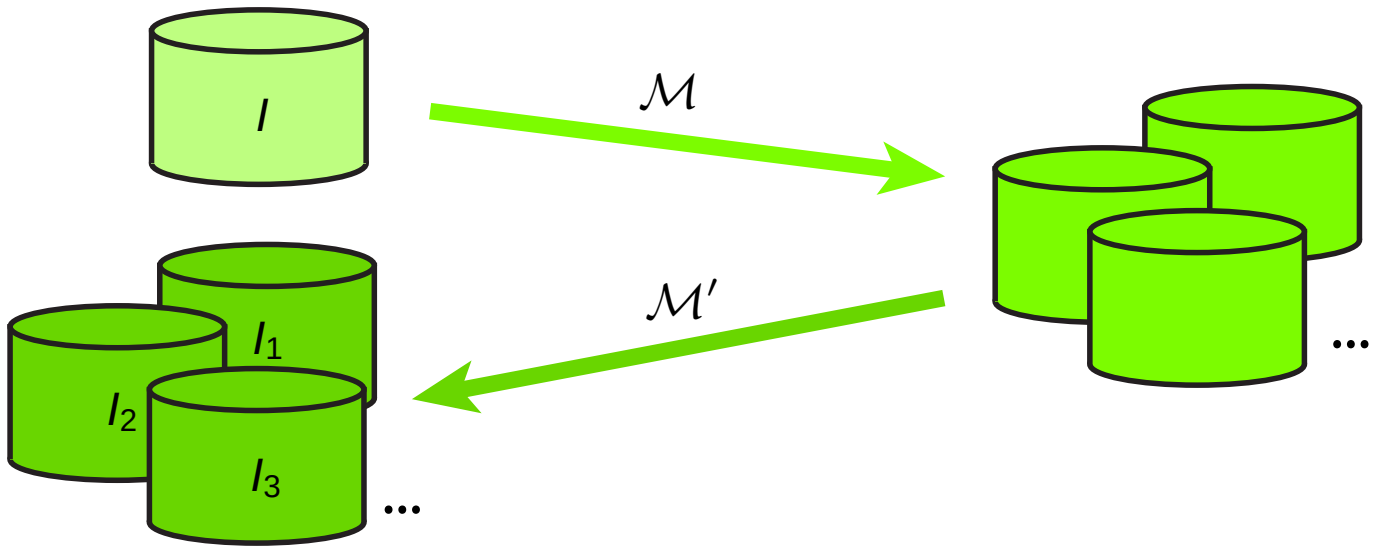
$$\underbrace{Q(I_1) \cap Q(I_2) \cap Q(I_3) \cap \cdots}_{\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)}$$

A mapping is a *Q-recovery* of $\mathcal{M}$ if it recovers sound information wrt Q

Definition

Given a source query Q, we say that a mapping $\mathcal{M}'$ is a *Q-recovery* of $\mathcal{M}$ iff for every source instance I:

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



$$\underbrace{Q(I_1) \cap Q(I_2) \cap Q(I_3) \cap \cdots}_{\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)} \quad \subseteq \quad Q(I)$$

We can extend the definition of Q-recovery to a class of queries $\mathcal{C}$

We can extend the definition of Q-recovery to a class of queries $\mathcal{C}$

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* $Q \in \mathcal{C}$.

We can extend the definition of Q-recovery to a class of queries $\mathcal{C}$

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:

We can extend the definition of Q-recovery to a class of queries $\mathcal{C}$

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*.

We can extend the definition of Q-recovery to a class of queries $\mathcal{C}$

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*.

We want to recover the maximum amount of information with respect to $\mathcal{C}$

We want to recover the maximum amount of information with respect to $\mathcal{C}$

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

We want to recover the maximum amount of information with respect to $\mathcal{C}$

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

We want to recover the maximum amount of information with respect to $\mathcal{C}$

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* than $\mathcal{M}_2$ as a $\mathcal{C}$-recovery of $\mathcal{M}$.

We want to recover the maximum amount of information with respect to $\mathcal{C}$

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* than $\mathcal{M}_2$ as a $\mathcal{C}$-recovery of $\mathcal{M}$.

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

A mapping is a $\mathcal{C}$-*maximum recovery* of $\mathcal{M}$

if it is the best $\mathcal{C}$-recovery of $\mathcal{M}$

A mapping is a $\mathcal{C}$-*maximum recovery* of $\mathcal{M}$ if it is the best $\mathcal{C}$-recovery of $\mathcal{M}$

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}_1$ is a $\mathcal{C}$-*maximum recovery* of $\mathcal{M}$ if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that

A mapping is a $\mathcal{C}$-*maximum recovery* of $\mathcal{M}$ if it is the best $\mathcal{C}$-recovery of $\mathcal{M}$

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}_1$ is a $\mathcal{C}$-*maximum recovery* 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 $Q \in \mathcal{C}$.

A mapping is a $\mathcal{C}$-*maximum recovery* of $\mathcal{M}$ if it is the best $\mathcal{C}$-recovery of $\mathcal{M}$

Definition

Given a class of queries $\mathcal{C}$, we say that $\mathcal{M}_1$ is a $\mathcal{C}$-*maximum recovery* 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

Previous notions of inverse correspond to particular classes of queries

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

Previous notions of inverse correspond to particular classes of queries

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

Theorem

- If $\mathcal{M}'$ is a *Fagin-inverse* of $\mathcal{M}$ then $\mathcal{M}'$ is a **UCQ**$^{\neq}$-maximum recovery of $\mathcal{M}$.

Previous notions of inverse correspond to particular classes of queries

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

Theorem

- If $\mathcal{M}'$ is a *Fagin-inverse* of $\mathcal{M}$ then $\mathcal{M}'$ is a **UCQ**$^{\neq}$-maximum recovery of $\mathcal{M}$.
- If $\mathcal{M}'$ is a *maximum recovery* of $\mathcal{M}$ then $\mathcal{M}'$ is an **All**-maximum recovery of $\mathcal{M}$.

We can use different classes of queries

to find more practical and useful inverses

We can use different classes of queries to find more practical and useful inverses

We know that:

- disjunctions make maximum recoveries impractical.

We can use different classes of queries to find more practical and useful inverses

We know that:

- disjunctions make maximum recoveries impractical.
- st-tgds focus in conjunctive data.

We can use different classes of queries to find more practical and useful inverses

We know that:

- disjunctions make maximum recoveries impractical.
- st-tgds focus in conjunctive data.
- **CQ**-maximum recoveries recover sound information wrt conjunctive queries.

We can use different classes of queries to find more practical and useful inverses

We know that:

- disjunctions make maximum recoveries impractical.
- st-tgds focus in conjunctive data.
- **CQ**-maximum recoveries recover sound information wrt conjunctive queries.

Theorem

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

How can we compute a **CQ**-maximum recovery?

Computing a **CQ**-maximum recovery consists of two main steps

Computing a **CQ**-maximum recovery consists of two main steps

Algorithm

- Step 1: Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.

Computing a **CQ**-maximum recovery consists of two main steps

Algorithm

- Step 1: Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.
- Step 2: Eliminate disjunctions and equalities of $\mathcal{M}'$.
  - Step 2.1: Eliminate equalities.
  - Step 2.2: Eliminate disjunctions.

Step 1: Compute a maximum recovery

Step 1: Compute a maximum recovery

Step 1:

- Let $\mathcal{M}$ be a mapping specified by st-tgds.

Step 1: Compute a maximum recovery

Step 1:

- Let $\mathcal{M}$ be a mapping specified by st-tgds.
- Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.

Step 1: Compute a maximum recovery

Step 1:

- Let $\mathcal{M}$ be a mapping specified by st-tgds.
- Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.
- $\mathcal{M}'$ is a set of **CQ$^{\mathbf{C}(\cdot)}$-to-UCQ$^{=}$** dependencies.

Step 1: Compute a maximum recovery

Step 1:

- Let $\mathcal{M}$ be a mapping specified by st-tgds.
- Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.
- $\mathcal{M}'$ is a set of **CQ**$^{\mathbf{C}(\cdot)}$**-to-UCQ**$^{=}$ dependencies.

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

Step 1: Compute a maximum recovery

Step 1:

- Let $\mathcal{M}$ be a mapping specified by st-tgds.
- Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.
- $\mathcal{M}'$ is a set of **CQ**$^{\mathbf{C}(\cdot)}$**-to-UCQ**$^{=}$ dependencies.

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

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

Step 2.1: Eliminate right-hand side equalities

Step 2.1: Eliminate right-hand side equalities

Example

$$A(x,y) \longrightarrow B(x,y) \vee (C(x) \wedge x = y)$$

Step 2.1: Eliminate right-hand side equalities

Example

$$
\begin{array}{lcll}
A(x,y) & \longrightarrow & B(x,y) & \vee \quad (C(x) \wedge x = y) \\
 & \Downarrow & & \\
A(x,y) \wedge x \neq y & \longrightarrow & B(x,y) &
\end{array}
$$

Step 2.1: Eliminate right-hand side equalities

Example

$$
\begin{array}{rcll}
A(x,y) & \longrightarrow & B(x,y) \quad \vee & (C(x) \wedge x = y) \\
 & \Downarrow & & \\
A(x,y) \wedge x \neq y & \longrightarrow & B(x,y) & \\
A(x,x) & \longrightarrow & B(x,x) \quad \vee & C(x)
\end{array}
$$

Step 2.1: Eliminate right-hand side equalities

Example

$$
\begin{array}{rcll}
A(x,y) & \longrightarrow & B(x,y) \quad \vee & (C(x) \wedge x = y) \\
 & \Downarrow & & \\
A(x,y) \wedge x \neq y & \longrightarrow & B(x,y) & \\
A(x,x) & \longrightarrow & B(x,x) \quad \vee & C(x)
\end{array}
$$

Step 2.1

- Let $\mathcal{M}'$ be the mapping constructed in Step 1.

Step 2.1: Eliminate right-hand side equalities

Example

$$
\begin{array}{rcll}
A(x,y) & \longrightarrow & B(x,y) & \vee \quad (C(x) \wedge x = y) \\
 & \Downarrow & & \\
A(x,y) \wedge x \neq y & \longrightarrow & B(x,y) & \\
A(x,x) & \longrightarrow & B(x,x) & \vee \quad C(x)
\end{array}
$$

Step 2.1

- 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 2.1: Eliminate right-hand side equalities

Example

$$
\begin{array}{rcll}
A(x,y) & \longrightarrow & B(x,y) & \vee \quad (C(x) \wedge x = y) \\
 & \Downarrow & & \\
A(x,y) \wedge x \neq y & \longrightarrow & B(x,y) & \\
A(x,x) & \longrightarrow & B(x,x) & \vee \quad C(x)
\end{array}
$$

Step 2.1

- 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.
- $\mathcal{M}''$ is a set of **CQ**$^{\mathbf{C}(\cdot),\neq}$**-to-UCQ** dependencies.

Step 2.1: Eliminate right-hand side equalities

Example

$$
\begin{array}{rcll}
A(x,y) & \longrightarrow & B(x,y) & \vee \quad (C(x) \wedge x = y) \\
 & \Downarrow & & \\
A(x,y) \wedge x \neq y & \longrightarrow & B(x,y) & \\
A(x,x) & \longrightarrow & B(x,x) & \vee \quad C(x)
\end{array}
$$

Step 2.1

- 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.
- $\mathcal{M}''$ is a set of **CQ**$^{\mathbf{C}(\cdot),\neq}$**-to-UCQ** dependencies.

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

Key concepts in Step 2.2

Key concepts in Step 2.2

- We have the following rule:

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

Key concepts in Step 2.2

- We have the following rule:

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

- We want to eliminate disjunctions, changing the rule to:

$$\varphi(\bar{x}) \longrightarrow \beta(\bar{x})$$

Key concepts in Step 2.2

- We have the following rule:

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

- We want to eliminate disjunctions, changing the rule to:

$$\varphi(\bar{x}) \longrightarrow \beta(\bar{x})$$

  such that β contains the *common conjunctive information* of $\{\beta_i\}$.

Key concepts in Step 2.2

- We have the following rule:

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

- We want to eliminate disjunctions, changing the rule to:

$$\varphi(\bar{x}) \longrightarrow \beta(\bar{x})$$

  such that β contains the *common conjunctive information* of $\{\beta_i\}$.

The key concepts in Step 2.2 are
homomorphisms and *Cartesian product of queries*.

Key concepts in Step 2.2: Homomorphisms

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

- maps existential variables to free or existential variables,

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

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

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

- maps existential variables to free or existential variables,
- is the identity over free variables,
- and *preserves* the relational structure of the query.

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

- maps existential variables to free or existential variables,
- is the identity over free variables,
- and *preserves* the relational structure of the query.

Example

$$\exists u \exists v\ A(x_1, u) \wedge B(v, v) \quad \xrightarrow{h} \quad A(x_1, x_2) \wedge B(x_1, x_1)$$

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

- maps existential variables to free or existential variables,
- is the identity over free variables,
- and *preserves* the relational structure of the query.

Example

$$\exists u \exists v\ A(x_1, u) \wedge B(v, v) \quad \xrightarrow[h(u) = x_2]{h} \quad A(x_1, x_2) \wedge B(x_1, x_1)$$

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

- maps existential variables to free or existential variables,
- is the identity over free variables,
- and *preserves* the relational structure of the query.

Example

$$\exists u \exists v\ A(x_1, u) \wedge B(v, v) \quad \xrightarrow{h} \quad A(x_1, x_2) \wedge B(x_1, x_1)$$

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

Key concepts in Step 2.2: Homomorphisms

Definition

A *homomorphism* between conjunctive queries is a function h that

- maps existential variables to free or existential variables,
- is the identity over free variables,
- and *preserves* the relational structure of the query.

Example

$$\exists u \exists v\, A(x_1, u) \wedge B(v, v) \quad \xrightarrow{h} \quad A(x_1, x_2) \wedge B(x_1, x_1)$$

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

Key concepts in Step 2.2:

Cartesian product of queries

Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

Q_1 Q_2

Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.

Q_1 Q_2

Q

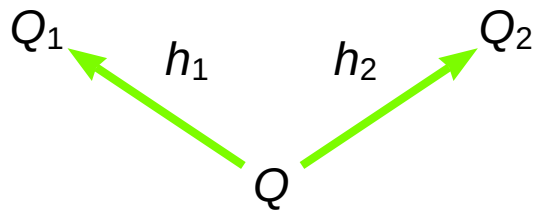
Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.



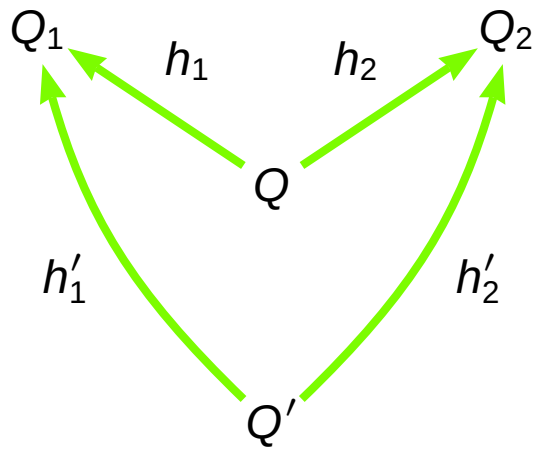
Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.



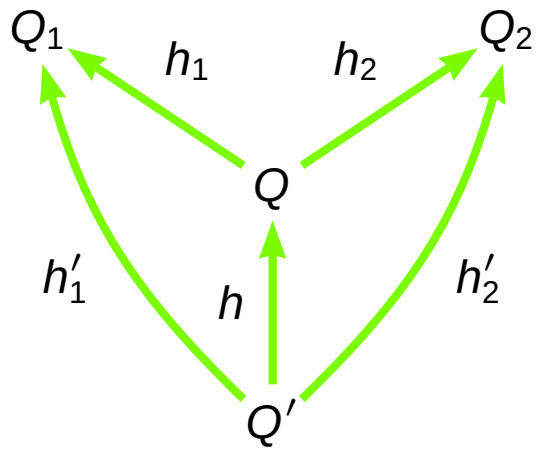
Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.



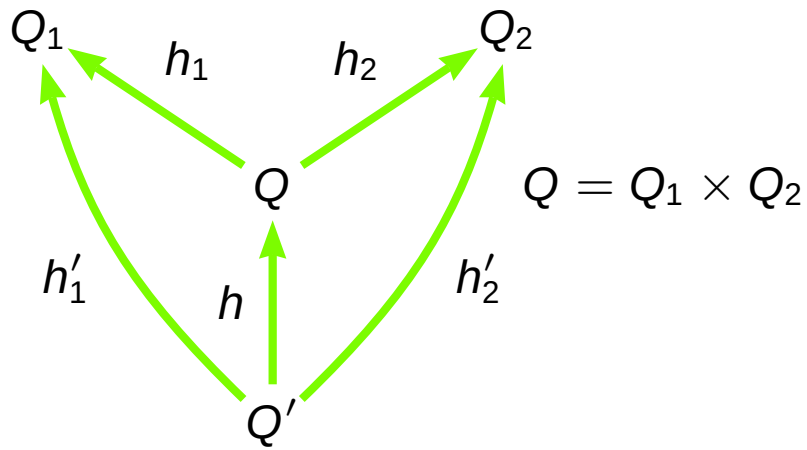
Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.



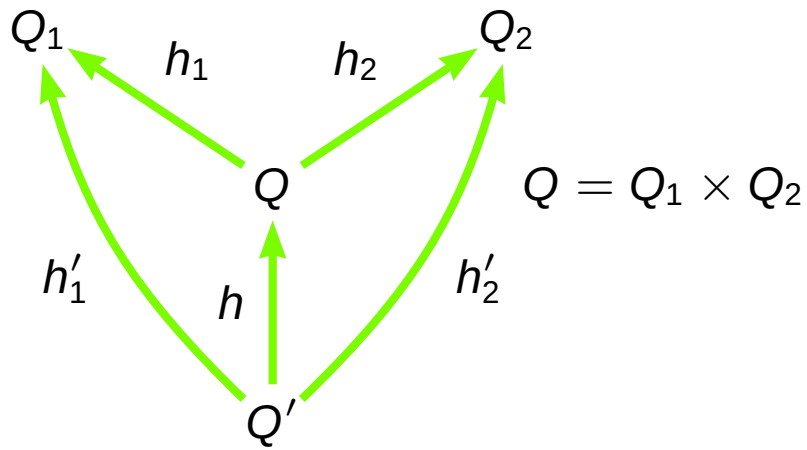
Key concepts in Step 2.2:
Cartesian product of queries

Definition *(semantic version)*

The conjunctive query Q is the *Cartesian product of* Q_1 and Q_2, or

$$Q_1 \times Q_2$$

if it is the *closest query* to both Q_1 and Q_2 in terms of homomorphism.



A simple extension of the Cartesian product of graphs.

Step 2.2: Eliminate disjunctions

Step 2.2: Eliminate disjunctions

Step 2.2:

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.1.

Step 2.2: Eliminate disjunctions

Step 2.2:

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.1.
- Construct $\mathcal{M}^\star$ by replacing every dependency

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

Step 2.2: Eliminate disjunctions

Step 2.2:

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.1.
- Construct $\mathcal{M}^\star$ by replacing every dependency

$$\begin{array}{ccc} \varphi(\bar{x}) & \longrightarrow & \beta_1(\bar{x}) \ \vee \ \beta_2(\bar{x}) \ \vee \cdots \ \vee \ \beta_n(\bar{x}) \\ & \Downarrow & \\ \varphi(\bar{x}) & \longrightarrow & \beta_1(\bar{x}) \ \times \ \beta_2(\bar{x}) \ \times \cdots \ \times \ \beta_n(\bar{x}) \end{array}$$

Step 2.2: Eliminate disjunctions

Step 2.2:

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.1.
- Construct $\mathcal{M}^\star$ by replacing every dependency

$$\begin{array}{ccc} \varphi(\bar{x}) & \longrightarrow & \beta_1(\bar{x}) \vee \beta_2(\bar{x}) \vee \cdots \vee \beta_n(\bar{x}) \\ & \Downarrow & \\ \varphi(\bar{x}) & \longrightarrow & \beta_1(\bar{x}) \times \beta_2(\bar{x}) \times \cdots \times \beta_n(\bar{x}) \end{array}$$

- $\mathcal{M}^\star$ is a set of **CQ**$^{\mathbf{C}(\cdot),\neq}$**-to-CQ** dependencies.

Step 2.2: Eliminate disjunctions

Step 2.2:

- Let $\mathcal{M}''$ be the mapping constructed in Step 2.1.
- Construct $\mathcal{M}^\star$ by replacing every dependency

$$\begin{array}{ccc} \varphi(\bar{x}) & \longrightarrow & \beta_1(\bar{x}) \vee \beta_2(\bar{x}) \vee \cdots \vee \beta_n(\bar{x}) \\ & \Downarrow & \\ \varphi(\bar{x}) & \longrightarrow & \beta_1(\bar{x}) \times \beta_2(\bar{x}) \times \cdots \times \beta_n(\bar{x}) \end{array}$$

- $\mathcal{M}^\star$ is a set of $\mathbf{CQ}^{\mathbf{c}(\cdot),\neq}$-**to**-**CQ** dependencies.

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

Summing up...

Algorithm

- Step 1: Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.
- Step 2: Eliminate disjunctions and equalities of $\mathcal{M}'$.
  - Step 2.1: Eliminate equalities using inequalities in the left side.
  - Step 2.2: Eliminate disjunctions using Cartesian product of queries.

Summing up...

Algorithm

- Step 1: Compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$.
- Step 2: Eliminate disjunctions and equalities of $\mathcal{M}'$.
  - Step 2.1: Eliminate equalities using inequalities in the left side.
  - Step 2.2: Eliminate disjunctions using Cartesian product of queries.

The output is a **CQ**-maximum recovery of $\mathcal{M}$
specified by st-tgds$^{\neq,\mathbf{C}(\cdot)}$.

st-tgds$^{\neq,\mathbf{C}}$ is the right language to specify an inverse

st-tgds$^{\neq,\mathbf{C}}$ is the right language to specify an inverse

Theorem

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

st-tgds$^{\neq,\mathbf{C}}$ is the right language to specify an inverse

Theorem

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

This language has the same good properties as st-tgds, in particular:

- the *chase* procedure can be used to exchange data,

st-tgds$^{\neq,\mathbf{C}}$ is the right language to specify an inverse

Theorem

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

This language has the same good properties as st-tgds, in particular:

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

Review

Review

- *Recovery:* recovers sound information.

Review

- *Recovery:* recovers sound information.
- *Maximum recovery:* recovers the maximum amount of sound information.

Review

- *Recovery:* recovers sound information.
- *Maximum recovery:* recovers the maximum amount of sound information.
- *Query rewriting* plays a key role to compute maximum recoveries.

Review

- *Recovery:* recovers sound information.
- *Maximum recovery:* recovers the maximum amount of sound information.
- *Query rewriting* plays a key role to compute maximum recoveries.
- $\mathcal{C}$*-recovery:* recovers sound information wrt a class of queries $\mathcal{C}$.

Review

- *Recovery:* recovers sound information.
- *Maximum recovery:* recovers the maximum amount of sound information.
- *Query rewriting* plays a key role to compute maximum recoveries.
- $\mathcal{C}$*-recovery:* recovers sound information wrt a class of queries $\mathcal{C}$.
- $\mathcal{C}$*-maximum recovery:* recovers the maximum amount of information wrt a class of queries $\mathcal{C}$.

Review

- *Recovery:* recovers sound information.
- *Maximum recovery:* recovers the maximum amount of sound information.
- *Query rewriting* plays a key role to compute maximum recoveries.
- $\mathcal{C}$*-recovery:* recovers sound information wrt a class of queries $\mathcal{C}$.
- $\mathcal{C}$*-maximum recovery:* recovers the maximum amount of information wrt a class of queries $\mathcal{C}$.
- *Cartesian product of queries* plays a key role to compute $\mathcal{C}$-maximum recoveries.

Conclusion

- *Maximum recovery* is the *best* that we can do to invert mappings.

Conclusion

- *Maximum recovery* is the *best* that we can do to invert mappings.
- *Certain answers* and *query rewriting* are old concepts that naturally arise when we study the inverse problem.

Conclusion

- *Maximum recovery* is the *best* that we can do to invert mappings.
- *Certain answers* and *query rewriting* are old concepts that naturally arise when we study the inverse problem.
- If we focus in certain kind of queries we can find more practical solutions to *recovering information in data exchange*.

Recovering information in data exchange

Cristian Riveros
Khipu Institute

Oxford University
Tue 26 Jan 2010