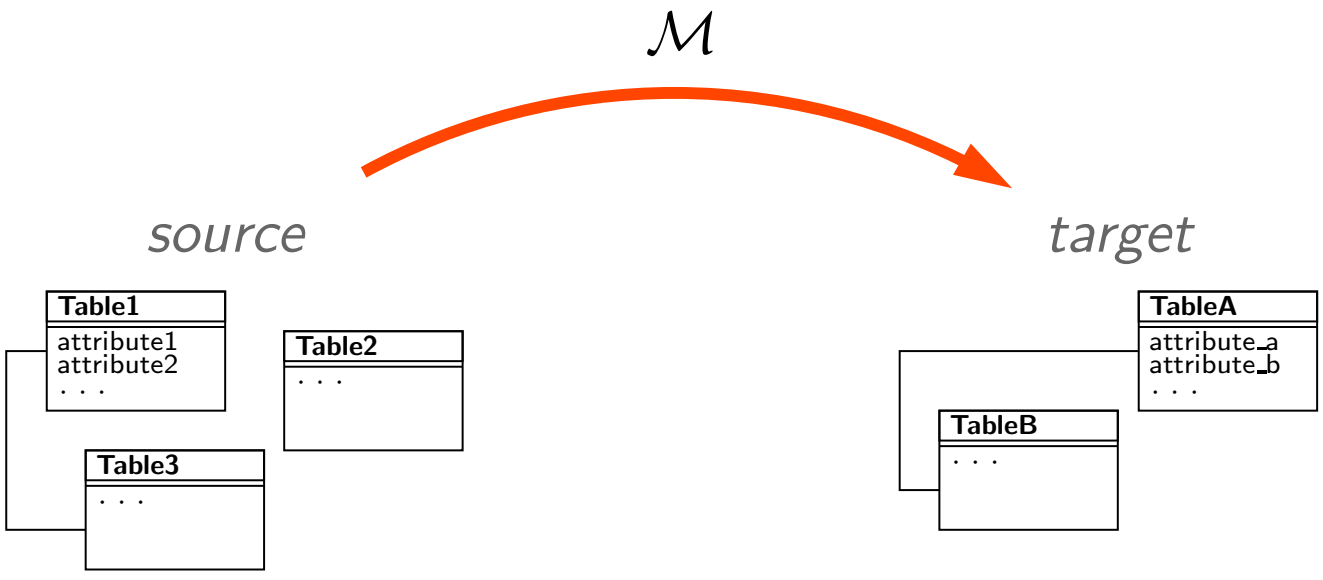
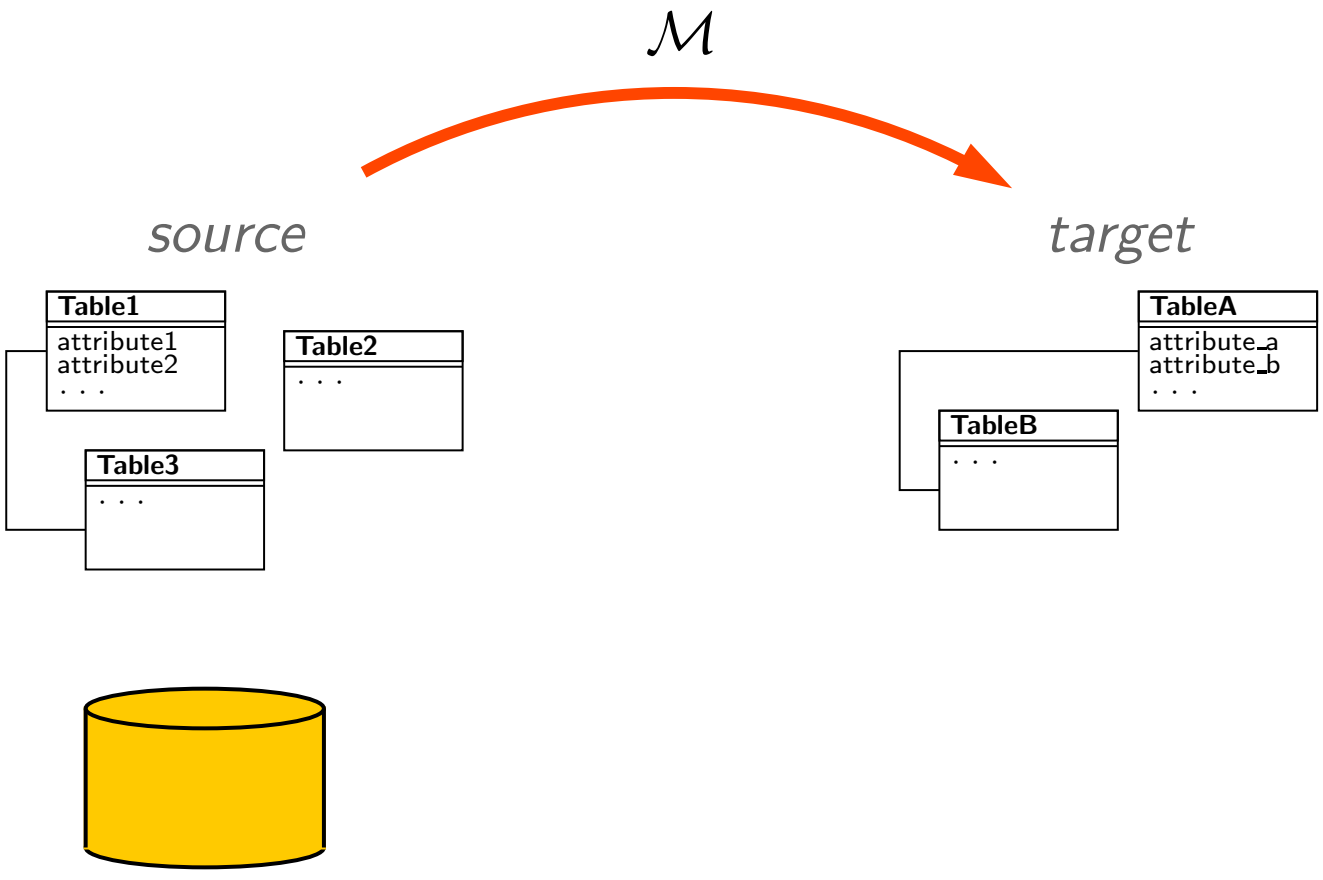
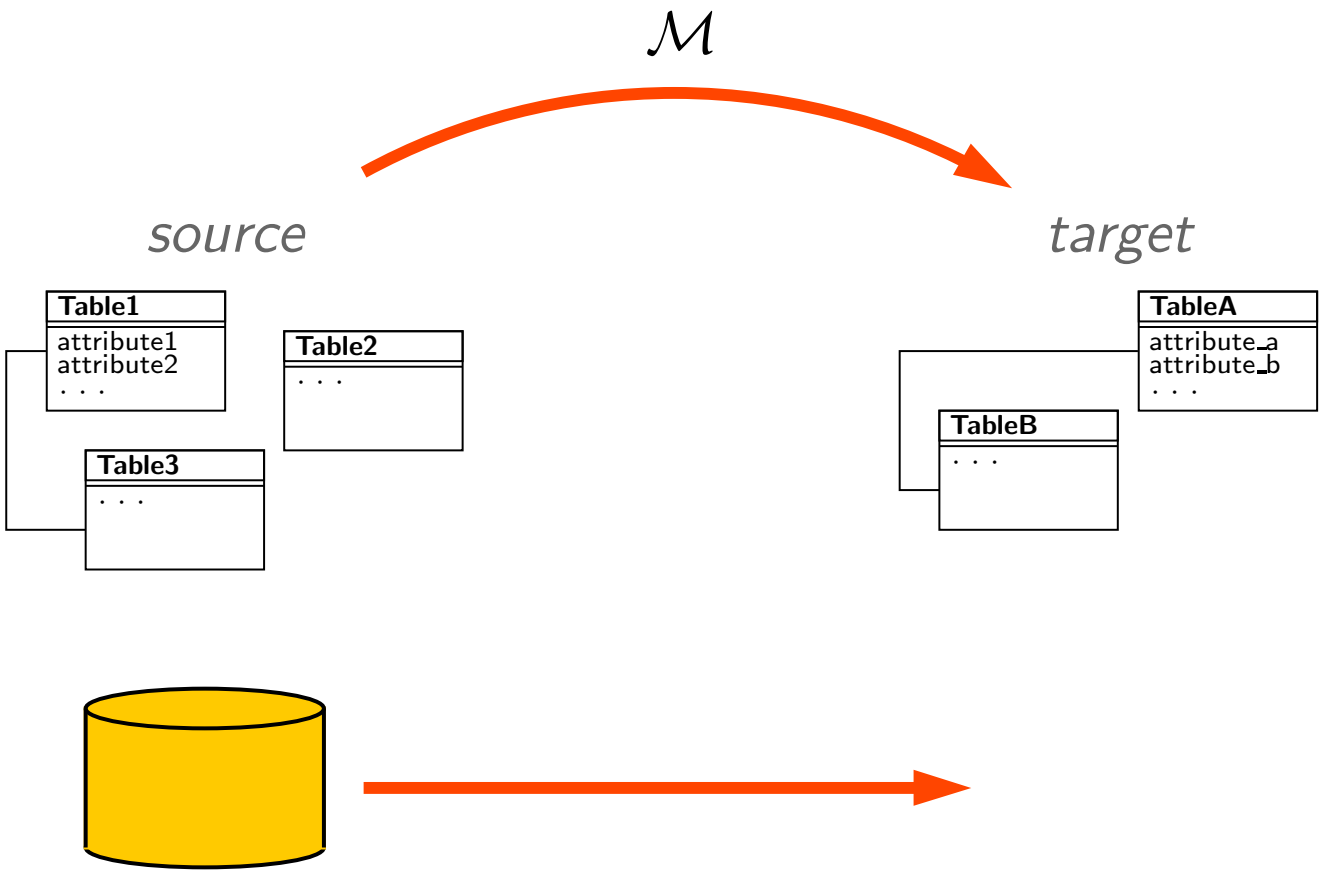
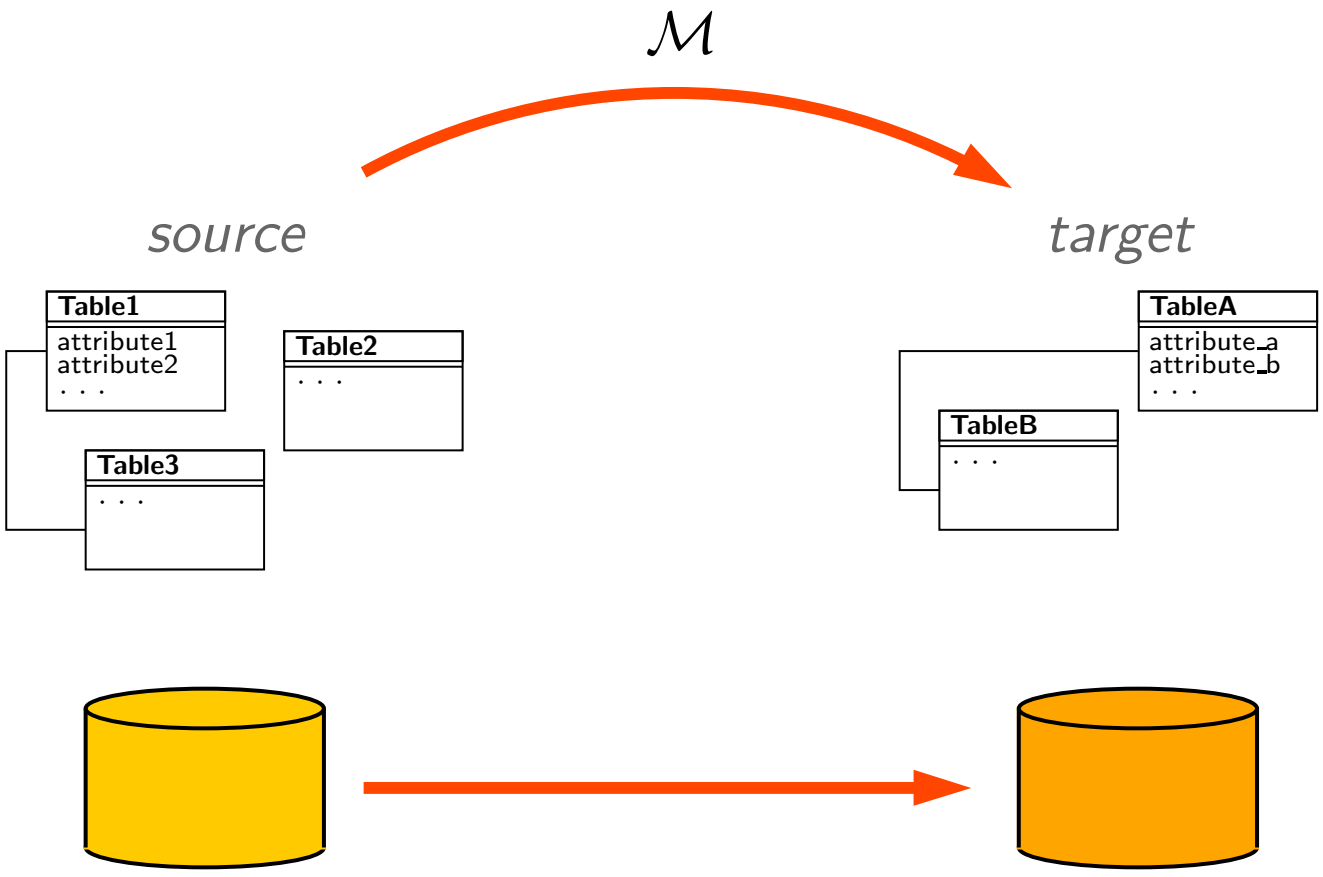
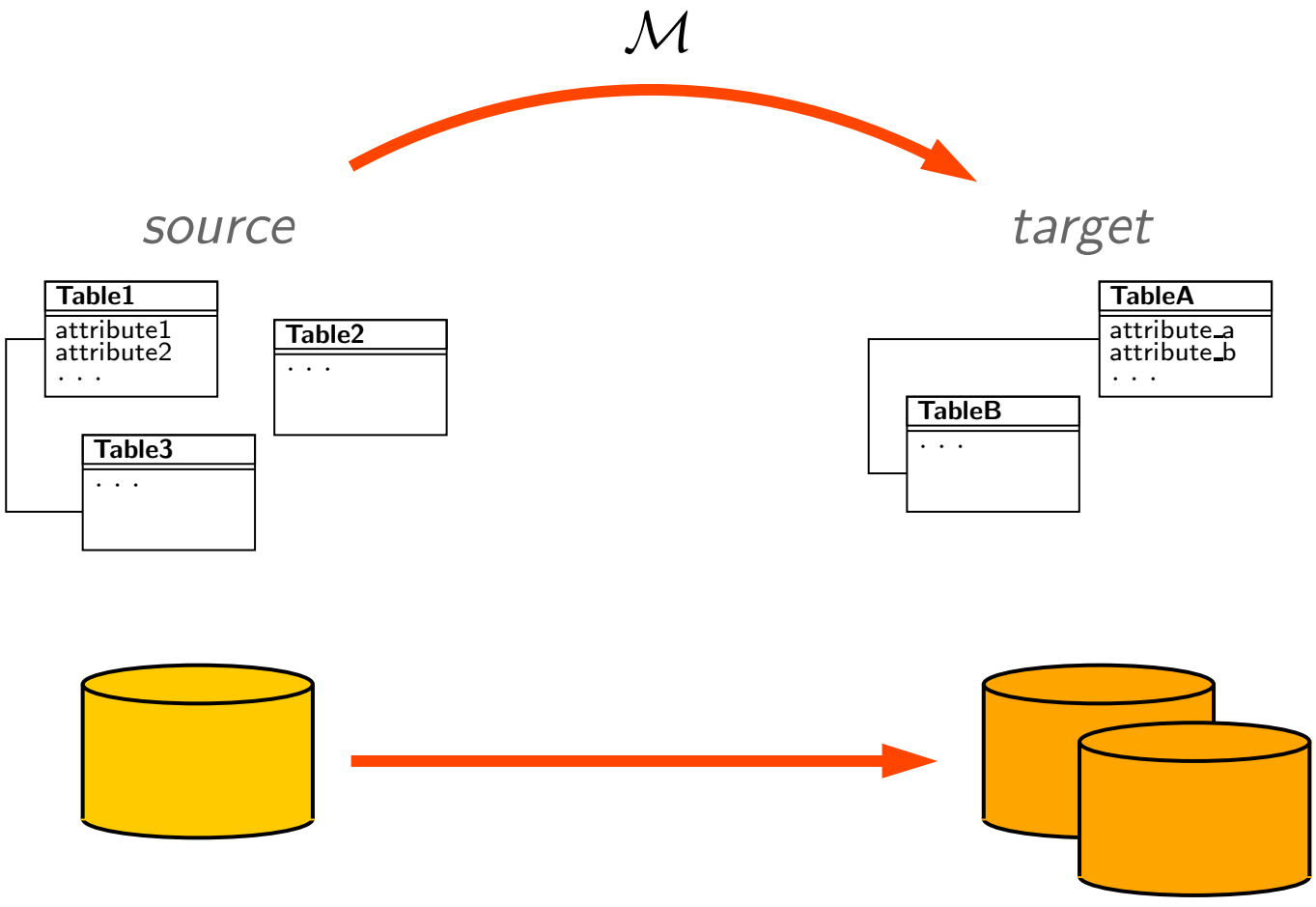
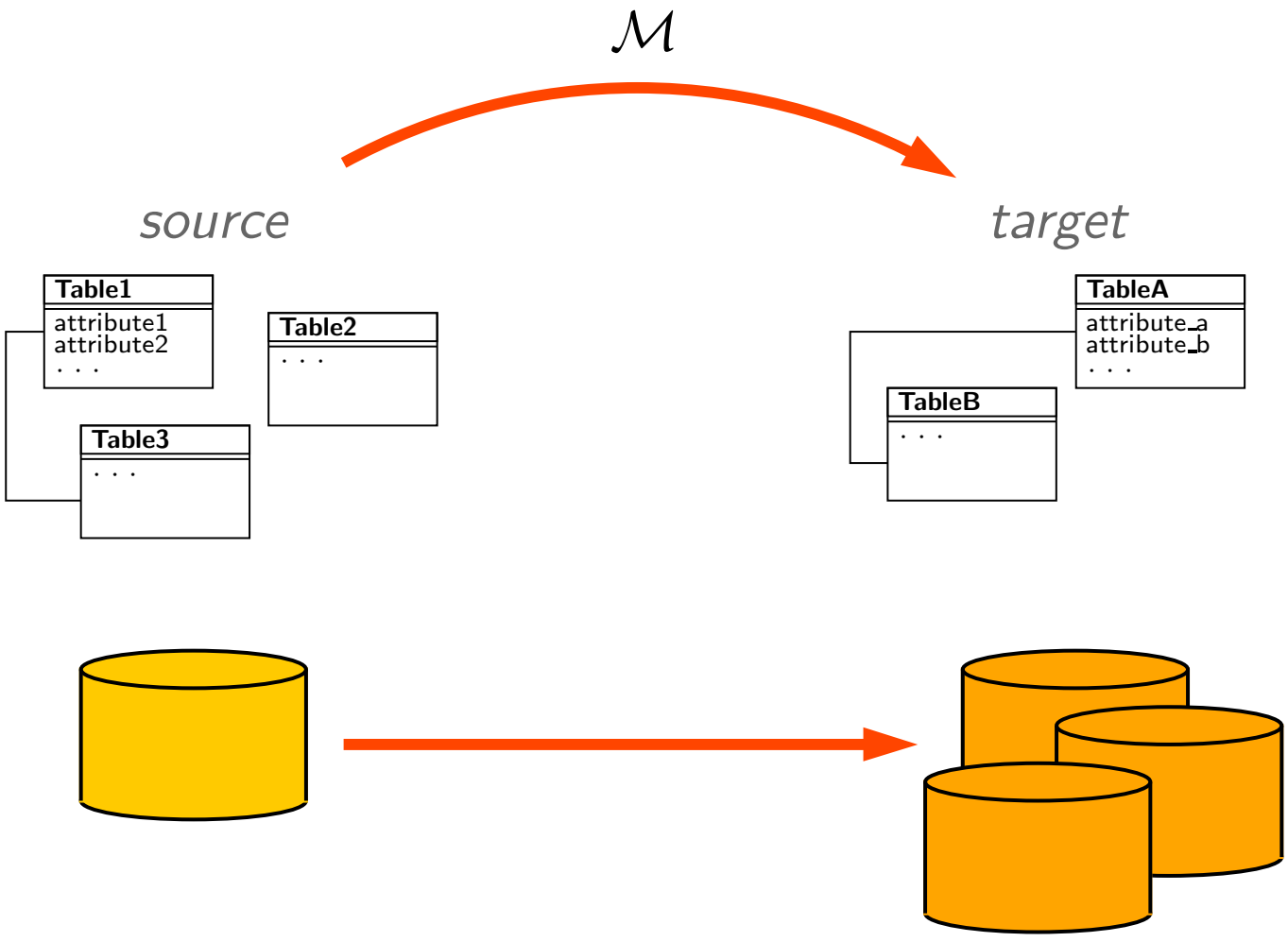

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .

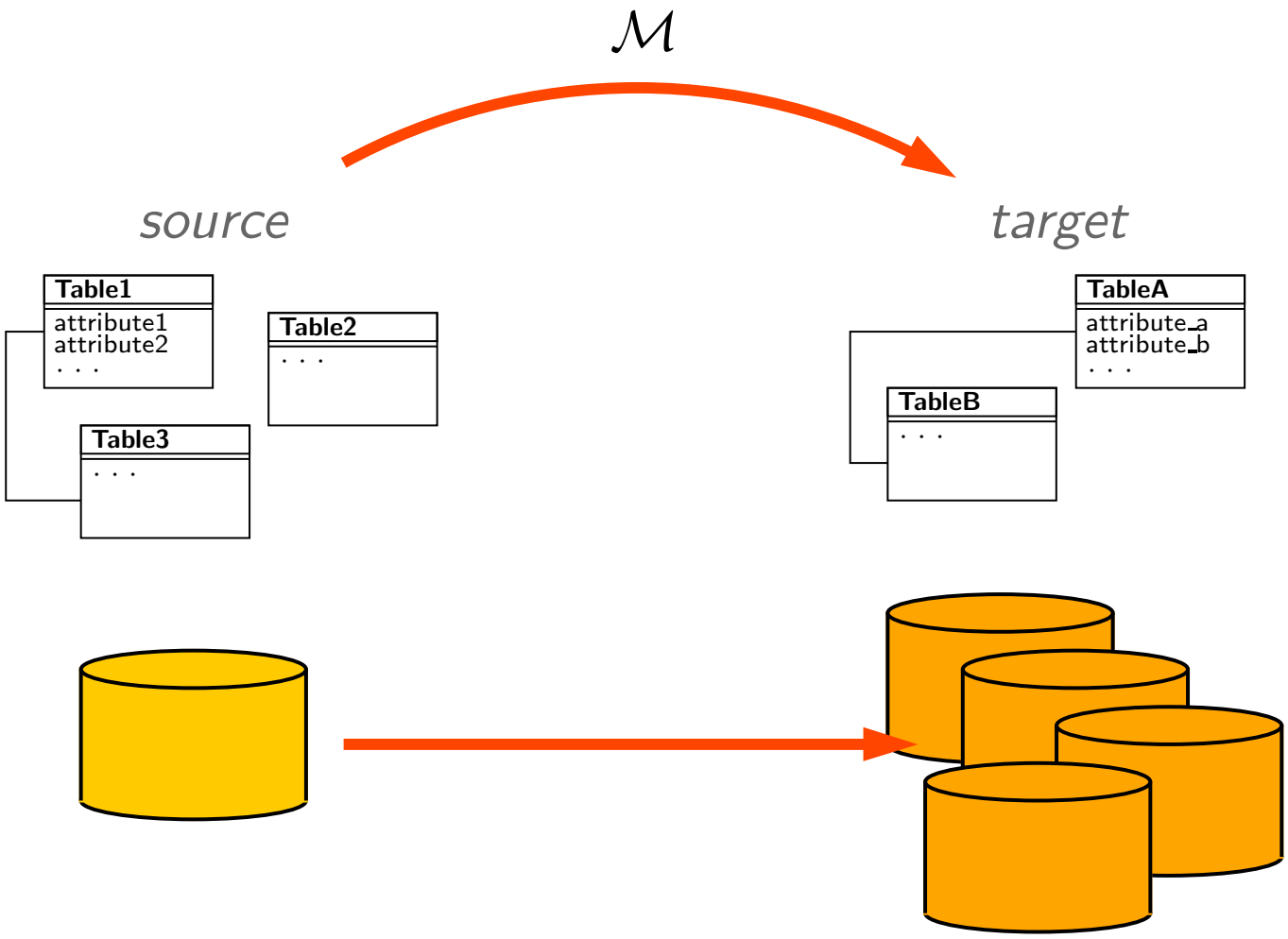

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .

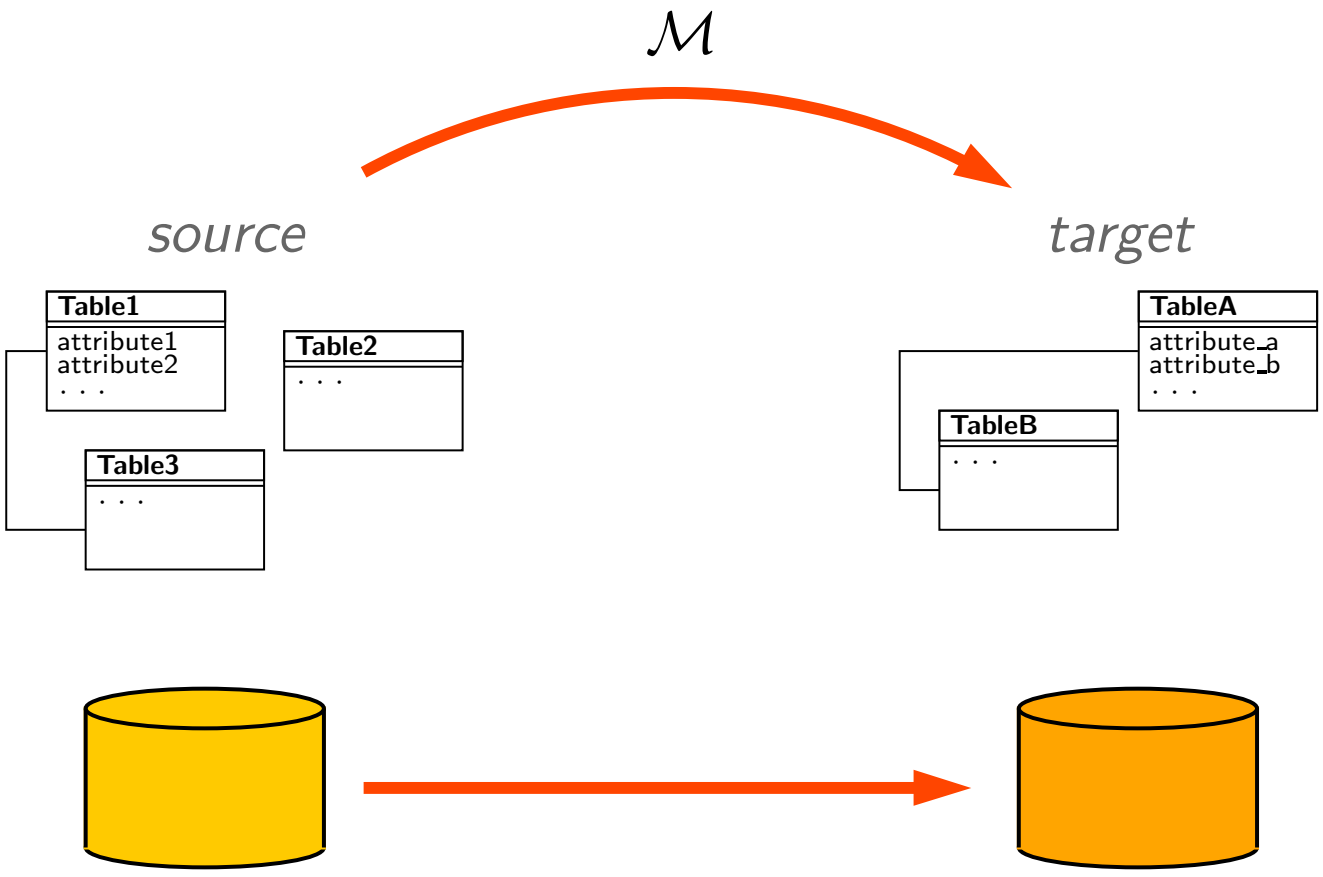

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .

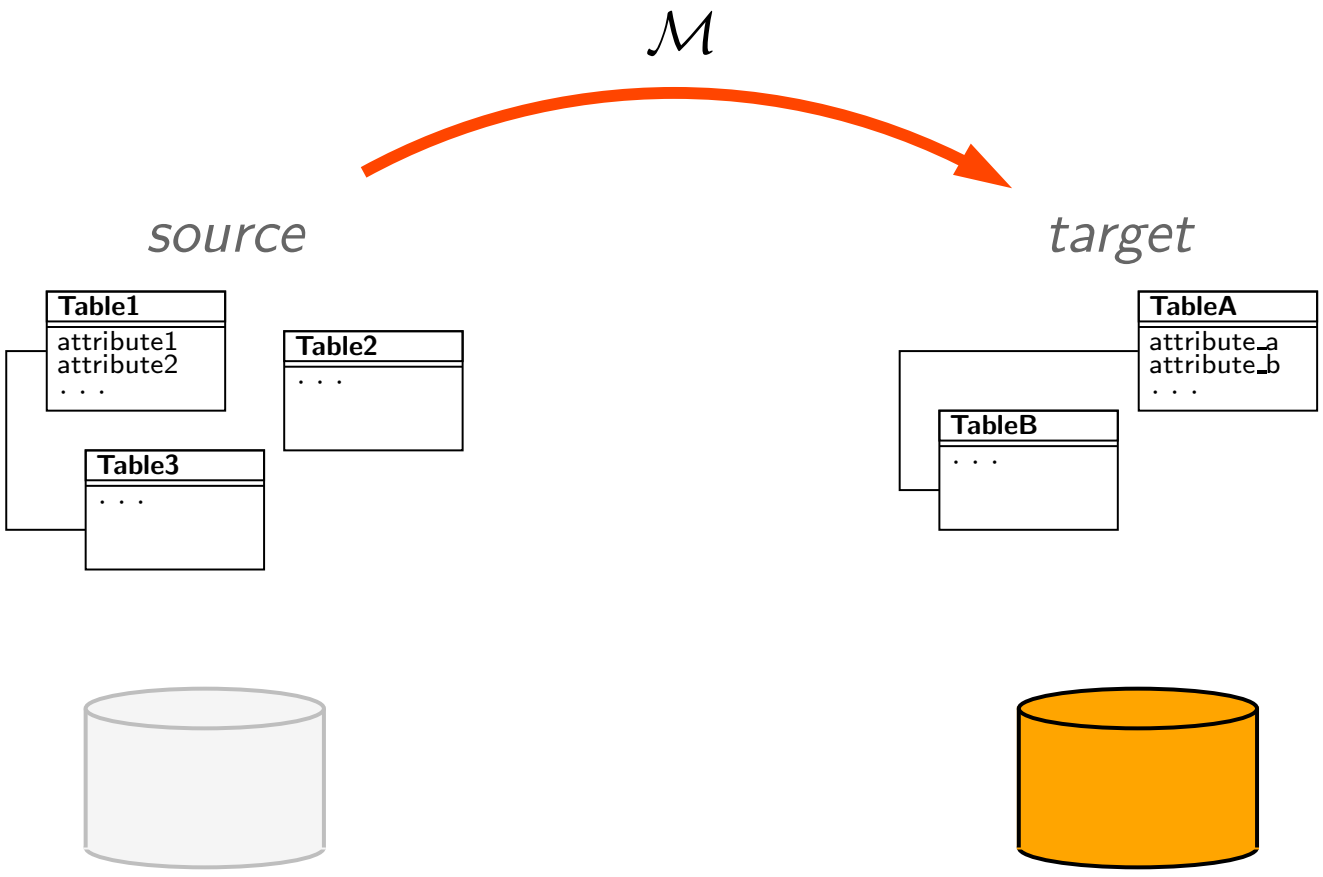

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .

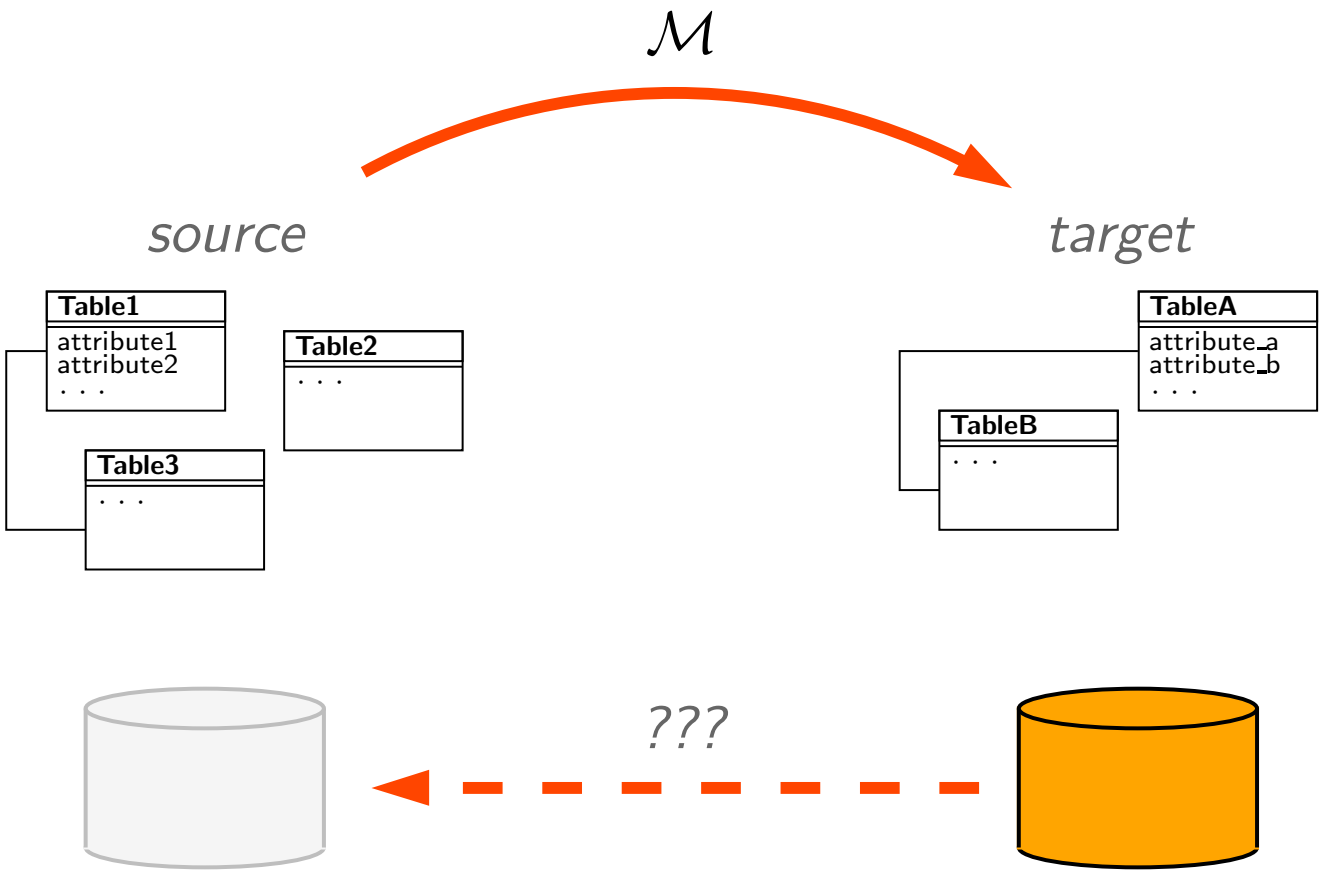

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .


$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .

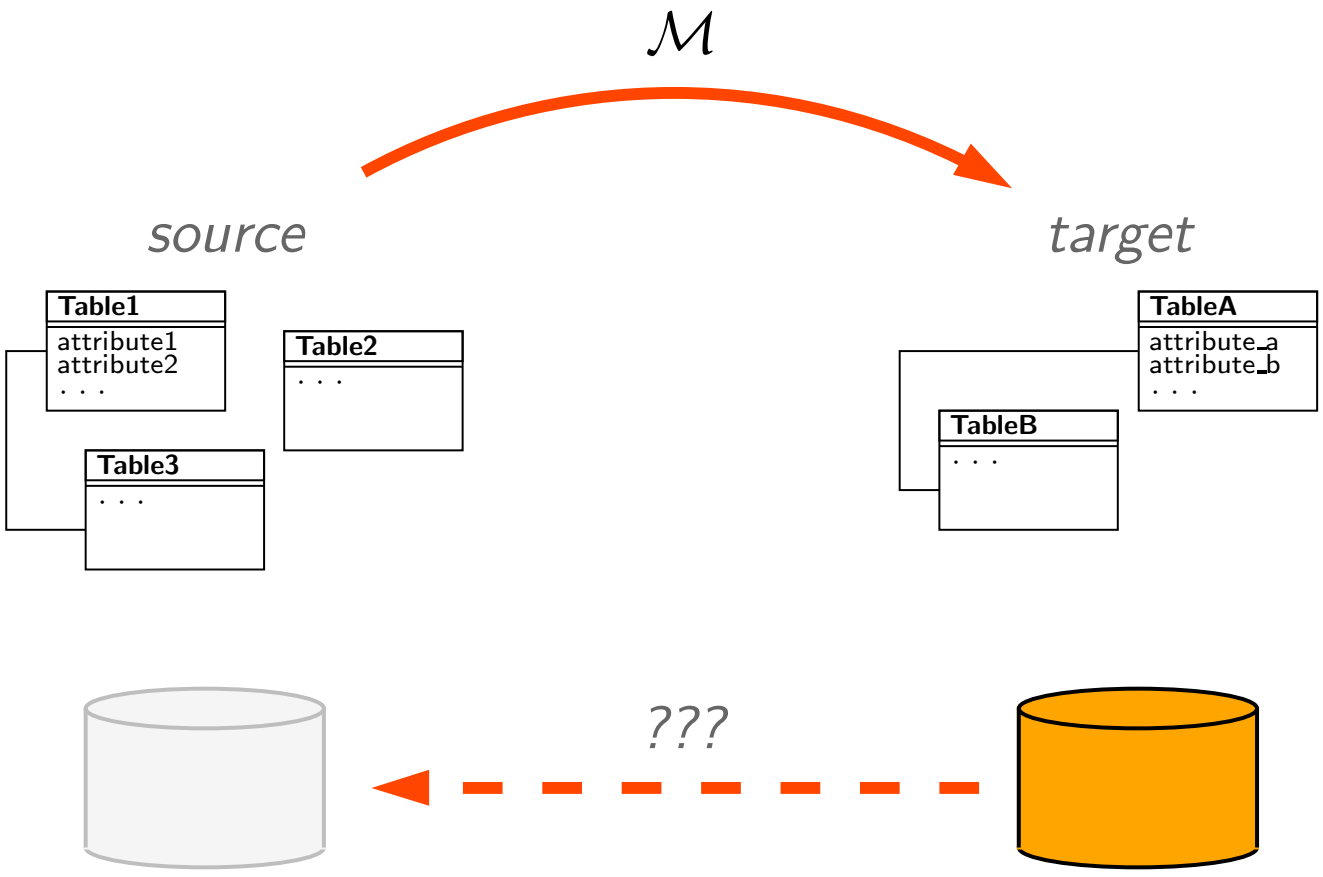

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .


$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .

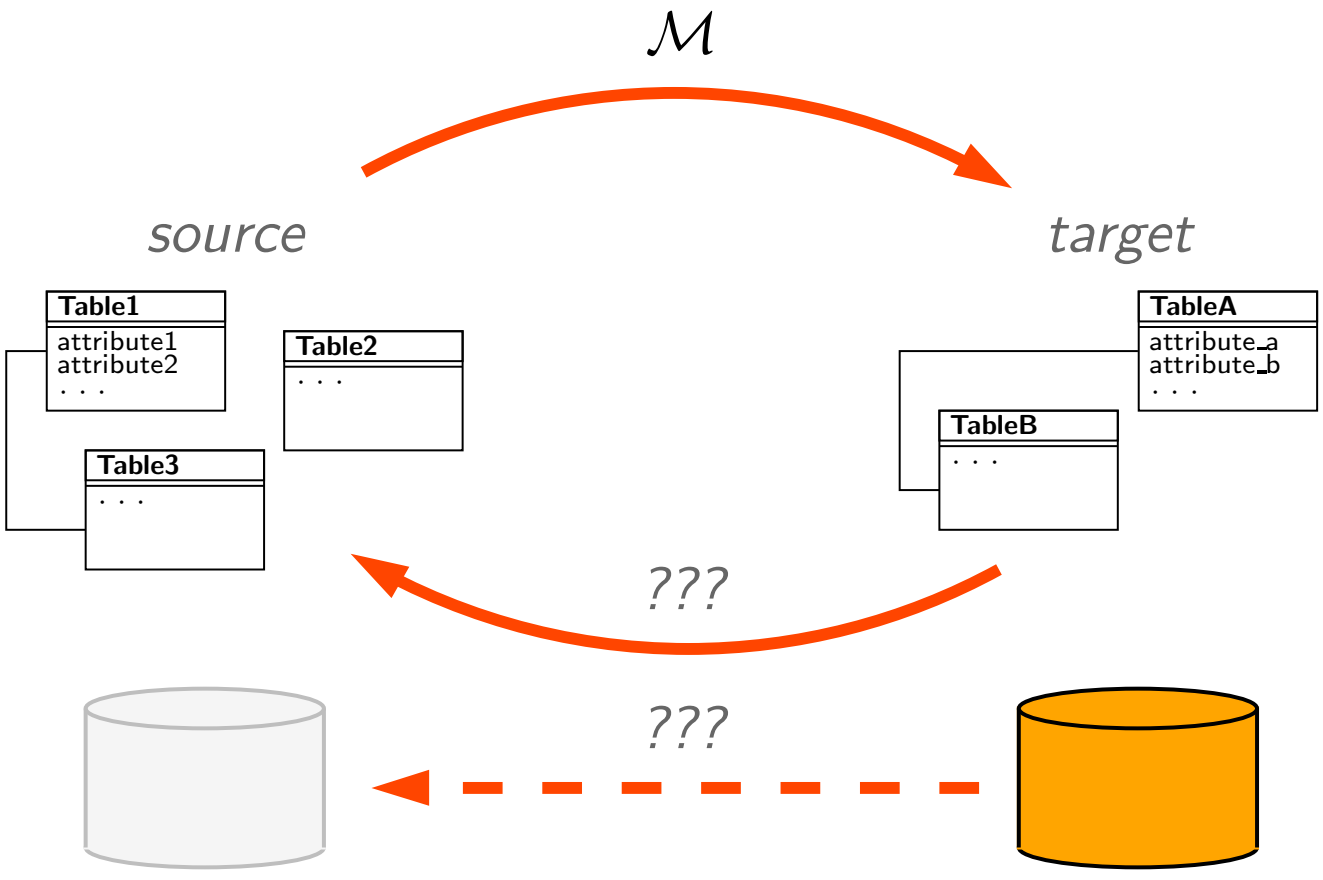

$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .


$\mathcal{M}$
source
target
Table1
attribute1
attribute2
. . .
Table2
. . .
Table3
. . .
TableA
attribute_a
attribute_b
. . .
TableB
. . .
???

How do we *recover* exchanged data?



How do we *recover* exchanged data?



We propose a *new* notion on
how to *reverse* a schema mapping.

We propose a *new* notion on
how to *reverse* a schema mapping.

Inverse [Fag06]:

- most mappings do not admit inverses [Fag06].

We propose a *new* notion on
how to *reverse* a schema mapping.

Inverse [Fag06]:

- most mappings do not admit inverses [Fag06].

Quasi-inverse [FKPT07] relaxes inverse:

- there are simple mappings with no quasi-inverse [FKPT07].

We propose a *new* notion on
how to *reverse* a schema mapping.

Inverse [Fag06]:

- most mappings do not admit inverses [Fag06].

Quasi-inverse [FKPT07] relaxes inverse:

- there are simple mappings with no quasi-inverse [FKPT07].

If (quasi-)inverses do not exist, what can we do?

We propose a *new* notion on
how to *reverse* a schema mapping.

Inverse [Fag06]:

- most mappings do not admit inverses [Fag06].

Quasi-inverse [FKPT07] relaxes inverse:

- there are simple mappings with no quasi-inverse [FKPT07].

If (quasi-)inverses do not exist, what can we do?

Question

Is there an alternative way
to recover (at least part of) the exchanged data?

The Recovery of a Schema Mapping: Bringing Exchanged Data Back

Marcelo Arenas Jorge Pérez Cristian Riveros

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Recovery: specifies how to recover *sound* information.

Recovery: specifies how to recover *sound* information.

Idea 1:

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

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

Emp(*name*, *lives_in*, *works_in*)

Shuttle(*name*, *destination*)

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

$\text{Emp}(\textit{name}, \textit{lives\_in}, \textit{works\_in})$ $\qquad$ $\text{Shuttle}(\textit{name}, \textit{destination})$

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \quad \longrightarrow \quad \text{Shuttle}(x, z)$$

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

Emp(*name*, *lives_in*, *works_in*) Shuttle(*name*, *destination*)

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \longrightarrow \text{Shuttle}(x, z)$$

$$\mathcal{M}_1: \quad \text{Shuttle}(x, z) \longrightarrow \exists U \exists V \, \text{Emp}(x, U, V)$$

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

$\mathsf{Emp}(\mathit{name}, \mathit{lives\_in}, \mathit{works\_in})$ $\qquad$ $\mathsf{Shuttle}(\mathit{name}, \mathit{destination})$

$$\mathcal{M}: \quad \mathsf{Emp}(x, y, z) \wedge y \neq z \longrightarrow \mathsf{Shuttle}(x, z)$$

$$\mathcal{M}_1: \quad \mathsf{Shuttle}(x, z) \longrightarrow \exists U \exists V\, \mathsf{Emp}(x, U, V) \quad \checkmark$$

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

Emp(*name*, *lives_in*, *works_in*) Shuttle(*name*, *destination*)

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \longrightarrow \text{Shuttle}(x, z)$$

$$\mathcal{M}_1: \quad \text{Shuttle}(x, z) \longrightarrow \exists U \exists V\ \text{Emp}(x, U, V) \quad \checkmark$$

$$\mathcal{M}_2: \quad \text{Shuttle}(x, z) \longrightarrow \exists U\ \text{Emp}(x, U, z)$$

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

$\text{Emp}(\textit{name}, \textit{lives\_in}, \textit{works\_in})$ $\qquad$ $\text{Shuttle}(\textit{name}, \textit{destination})$

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \longrightarrow \text{Shuttle}(x, z)$$

$$\mathcal{M}_1: \quad \text{Shuttle}(x, z) \longrightarrow \exists U \exists V\ \text{Emp}(x, U, V) \quad \checkmark$$

$$\mathcal{M}_2: \quad \text{Shuttle}(x, z) \longrightarrow \exists U\ \text{Emp}(x, U, z) \quad \checkmark$$

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

Emp(*name*, *lives_in*, *works_in*) Shuttle(*name*, *destination*)

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \longrightarrow \text{Shuttle}(x, z)$$

$$\begin{array}{lll} \mathcal{M}_1: & \text{Shuttle}(x, z) \longrightarrow & \exists U \exists V\ \text{Emp}(x, U, V) \quad \checkmark \\ \mathcal{M}_2: & \text{Shuttle}(x, z) \longrightarrow & \exists U\ \text{Emp}(x, U, z) \quad \checkmark \\ \mathcal{M}_3: & \text{Shuttle}(x, z) \longrightarrow & \exists V\ \text{Emp}(x, z, V) \end{array}$$

Recovery: specifies how to recover *sound* information.

Idea 1:

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

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

Example

$\text{Emp}(\textit{name}, \textit{lives\_in}, \textit{works\_in})$ $\qquad$ $\text{Shuttle}(\textit{name}, \textit{destination})$

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \longrightarrow \text{Shuttle}(x, z)$$

$$\begin{array}{lll} \mathcal{M}_1: & \text{Shuttle}(x, z) \longrightarrow & \exists U \exists V\ \text{Emp}(x, U, V) \quad \checkmark \\ \mathcal{M}_2: & \text{Shuttle}(x, z) \longrightarrow & \exists U\ \text{Emp}(x, U, z) \quad \checkmark \\ \mathcal{M}_3: & \text{Shuttle}(x, z) \longrightarrow & \exists V\ \text{Emp}(x, z, V) \quad \times \end{array}$$

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Example

$$\mathcal{M}: \quad \mathsf{Emp}(x, y, z) \wedge y \neq z \quad \longrightarrow \quad \mathsf{Shuttle}(x, z)$$

$$\begin{array}{lrcrl} \mathcal{M}_1: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U \exists V & \mathsf{Emp}(x, U, V) \\ \mathcal{M}_2: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U & \mathsf{Emp}(x, U, z) \end{array}$$

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Example

$$\mathcal{M}: \quad \mathsf{Emp}(x, y, z) \wedge y \neq z \quad \longrightarrow \quad \mathsf{Shuttle}(x, z)$$

$$\begin{array}{lrcl} \mathcal{M}_1: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U \exists V \quad \mathsf{Emp}(x, U, V) \\ \mathcal{M}_2: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U \quad \mathsf{Emp}(x, U, z) \end{array}$$

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

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Example

$$\mathcal{M}: \quad \mathsf{Emp}(x, y, z) \wedge y \neq z \quad \longrightarrow \quad \mathsf{Shuttle}(x, z)$$

$$\begin{array}{lllll}
\mathcal{M}_1: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U \exists V & \mathsf{Emp}(x, U, V) \\
\mathcal{M}_2: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U & \mathsf{Emp}(x, U, z) \\
\mathcal{M}_4: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U & \mathsf{Emp}(x, U, z) \wedge U \neq z
\end{array}$$

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

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Example

$$\mathcal{M}: \quad \mathsf{Emp}(x, y, z) \wedge y \neq z \quad \longrightarrow \quad \mathsf{Shuttle}(x, z)$$

$$\begin{array}{lllll}
\mathcal{M}_1: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U \exists V & \mathsf{Emp}(x, U, V) \\
\mathcal{M}_2: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U & \mathsf{Emp}(x, U, z) \\
\mathcal{M}_4: & \mathsf{Shuttle}(x, z) & \longrightarrow & \exists U & \mathsf{Emp}(x, U, z) \wedge U \neq z
\end{array}$$

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

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

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Example

$$\mathcal{M}: \quad \text{Emp}(x, y, z) \wedge y \neq z \quad \longrightarrow \quad \text{Shuttle}(x, z)$$

$$\begin{array}{lllll}
\mathcal{M}_1: & \text{Shuttle}(x, z) & \longrightarrow & \exists U \exists V & \text{Emp}(x, U, V) \\
\mathcal{M}_2: & \text{Shuttle}(x, z) & \longrightarrow & \exists U & \text{Emp}(x, U, z) \\
\mathcal{M}_4: & \text{Shuttle}(x, z) & \longrightarrow & \exists U & \text{Emp}(x, U, z) \wedge U \neq z
\end{array}$$

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

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

Idea 2:

- Choose a recovery $\mathcal{M}'$ of $\mathcal{M}$ that is better than every other.

Maximum recovery, the *most informative* recovery

Can we compare alternative recoveries?

Example

$$\mathcal{M}: \quad \text{Emp}(x,y,z) \wedge y \neq z \longrightarrow \text{Shuttle}(x,z)$$

$$\begin{array}{lllll}
\mathcal{M}_1: & \text{Shuttle}(x,z) & \longrightarrow & \exists U \exists V & \text{Emp}(x,U,V) \\
\mathcal{M}_2: & \text{Shuttle}(x,z) & \longrightarrow & \exists U & \text{Emp}(x,U,z) \\
\mathcal{M}_4: & \text{Shuttle}(x,z) & \longrightarrow & \exists U & \text{Emp}(x,U,z) \wedge U \neq z
\end{array}$$

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

Idea 2:

- Choose a recovery $\mathcal{M}'$ of $\mathcal{M}$ that is better than every other.

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

Our contributions

Conceptually:

- recovery and maximum recovery

Technically:

- characterization of the existence of a maximum recovery
- a positive result for the most common mappings
- comparison with (quasi-)inverse
- algorithms
- complexity study

Outline

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 $\mathcal{M}$

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

Definition

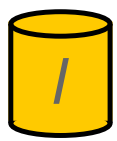
$\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ if for every source instance I

$$I \textit{ is a possible solution for itself} \text{ under } \mathcal{M} \circ \mathcal{M}'.$$

Definition

$\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ if for every source instance I

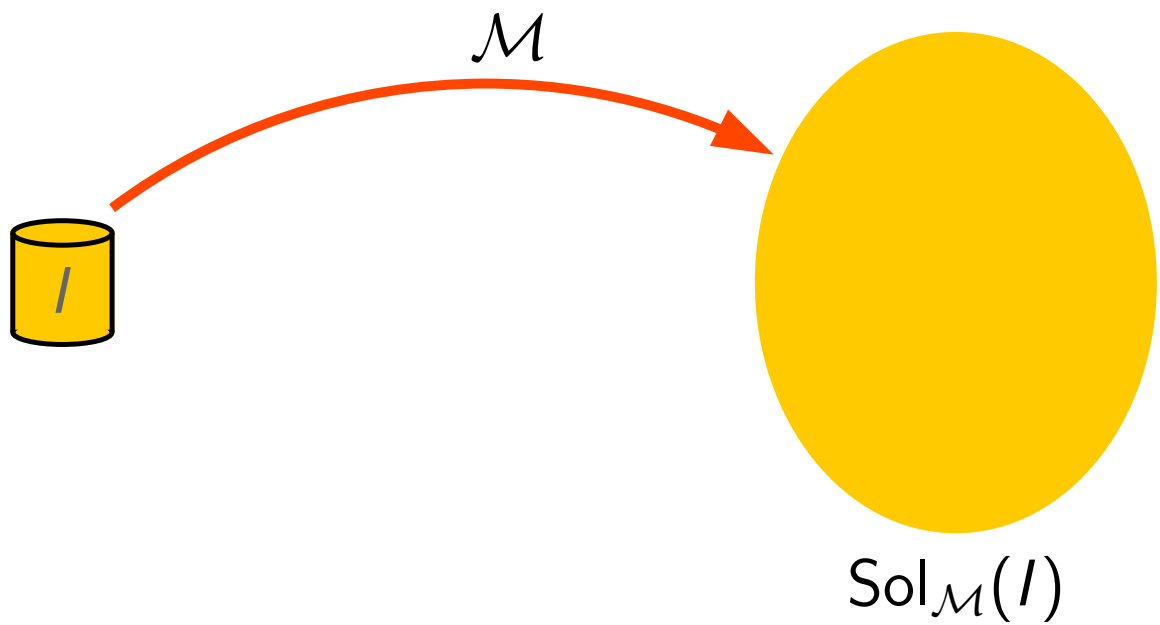
$$I \textit{ is a possible solution for itself} \text{ under } \mathcal{M} \circ \mathcal{M}'.$$



Definition

$\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ if for every source instance I

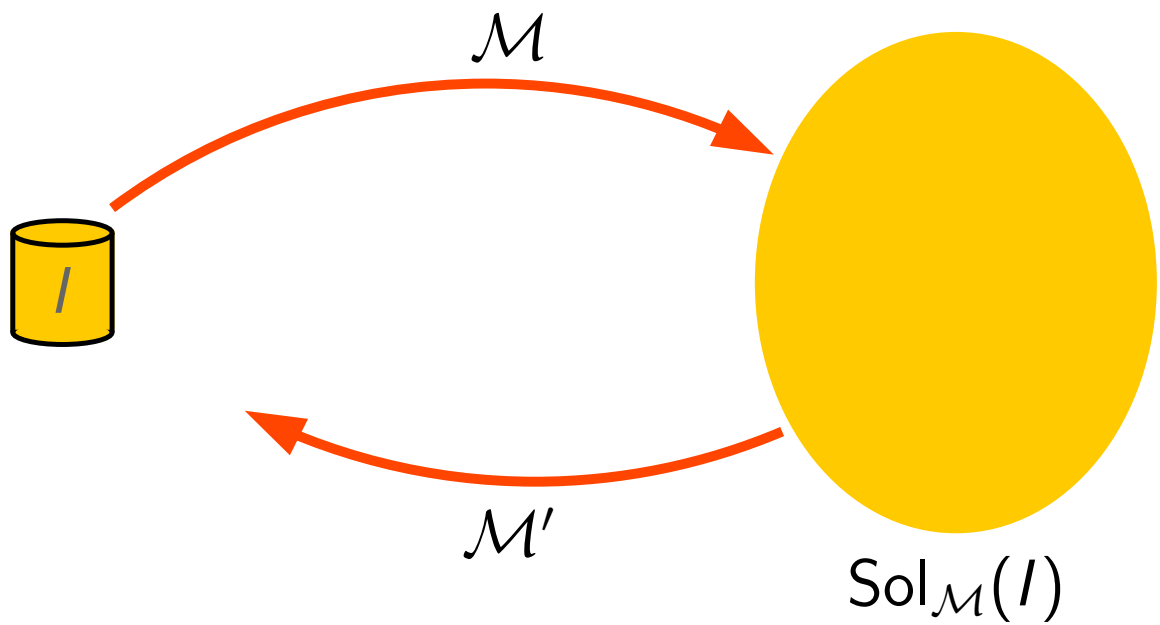
I *is a possible solution for itself* under $\mathcal{M} \circ \mathcal{M}'$.



Definition

$\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ if for every source instance I

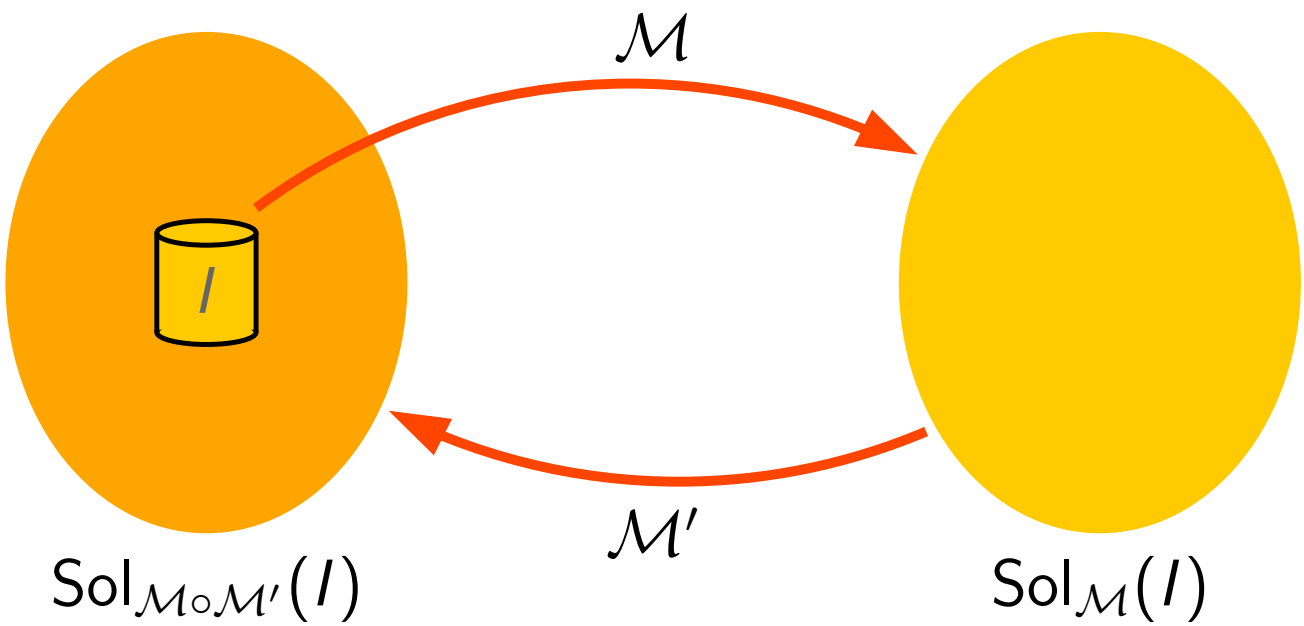
$$I \textit{ is a possible solution for itself} \text{ under } \mathcal{M} \circ \mathcal{M}'.$$

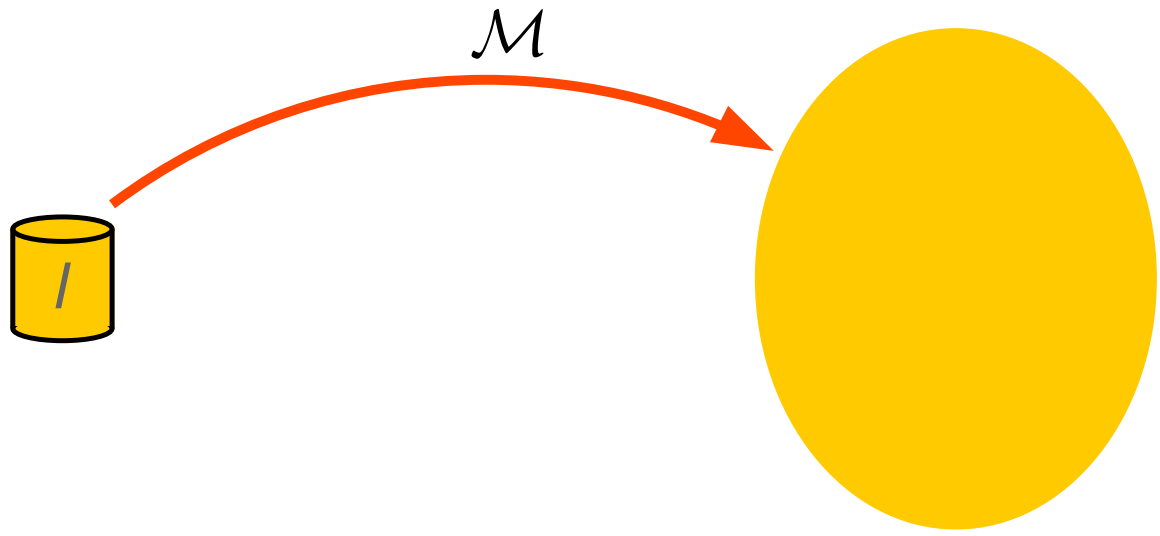


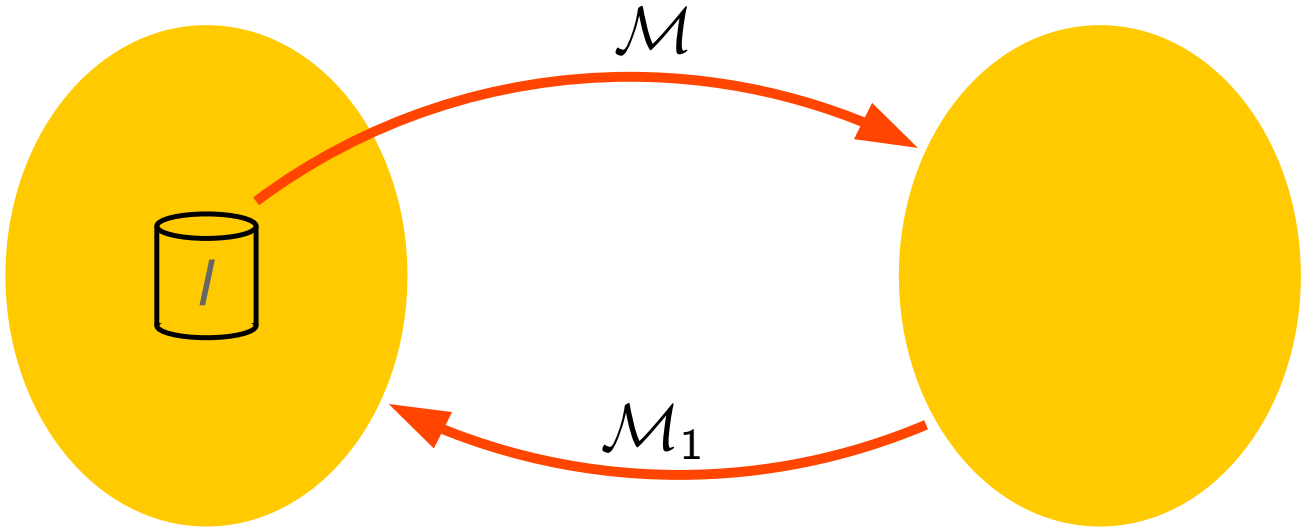
Definition

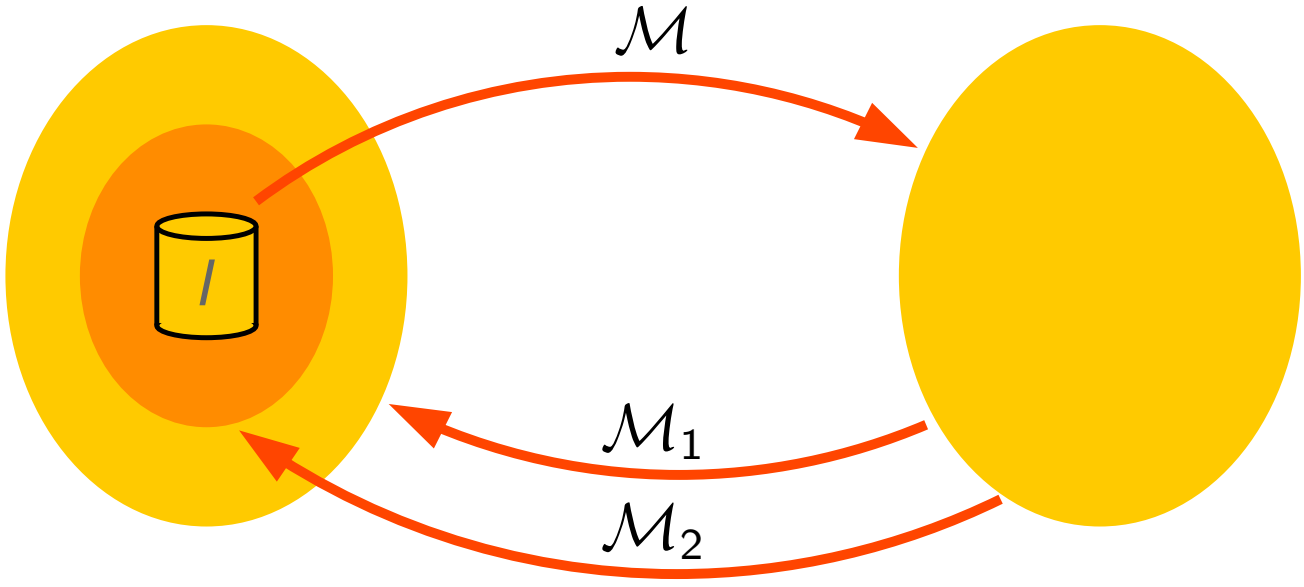
$\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ if for every source instance I

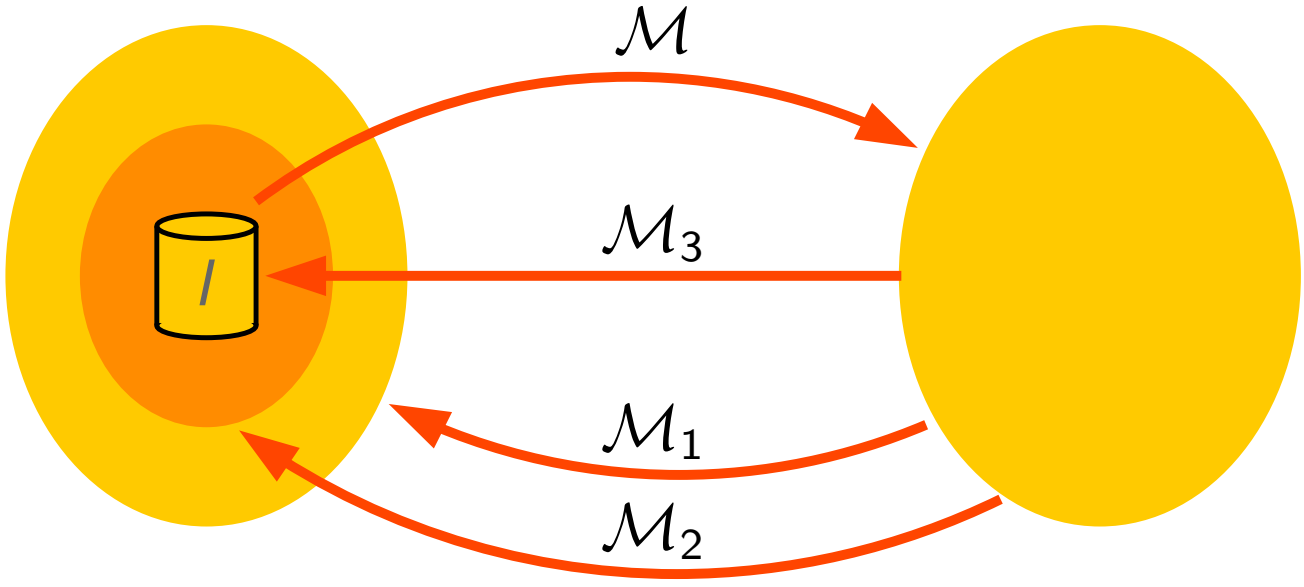
I *is a possible solution for itself* under $\mathcal{M} \circ \mathcal{M}'$.

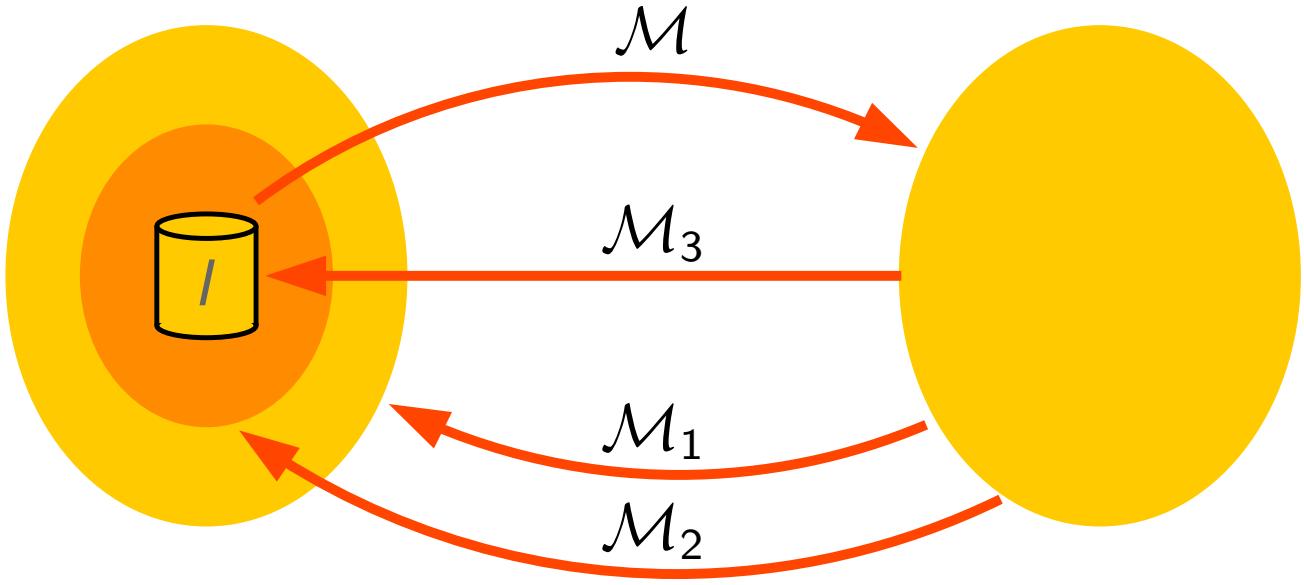



$\mathcal{M}$
I

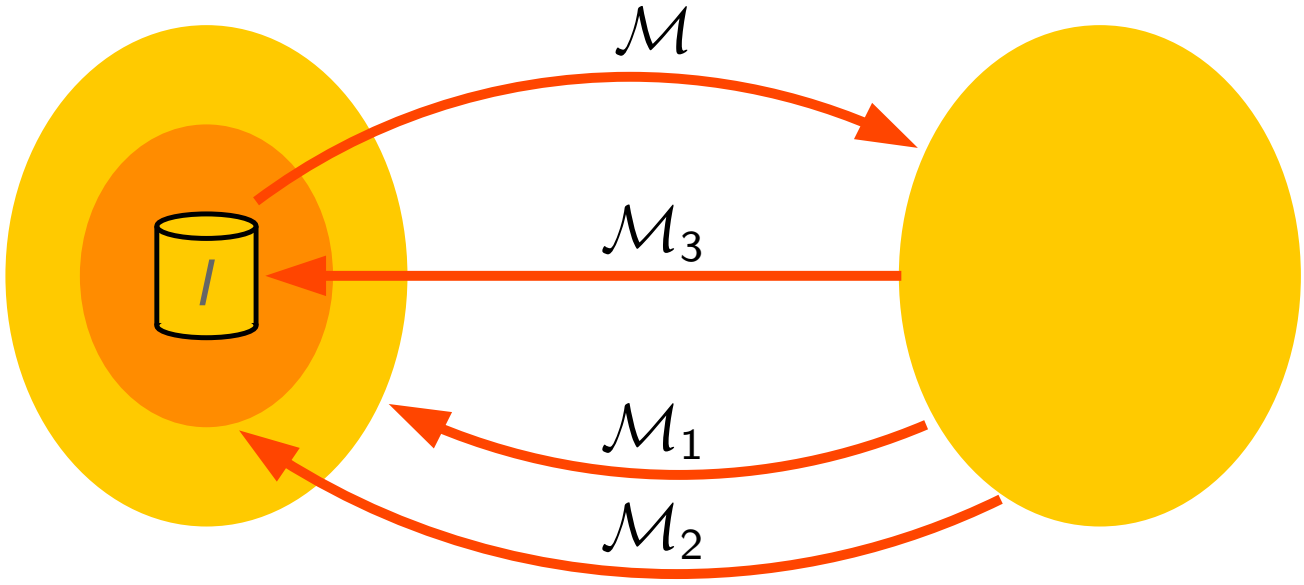

$\mathcal{M}$
I
$\mathcal{M}_1$


$\mathcal{M}$
I
$\mathcal{M}_1$
$\mathcal{M}_2$


$\mathcal{M}$
$\mathcal{M}_3$
I
$\mathcal{M}_1$
$\mathcal{M}_2$

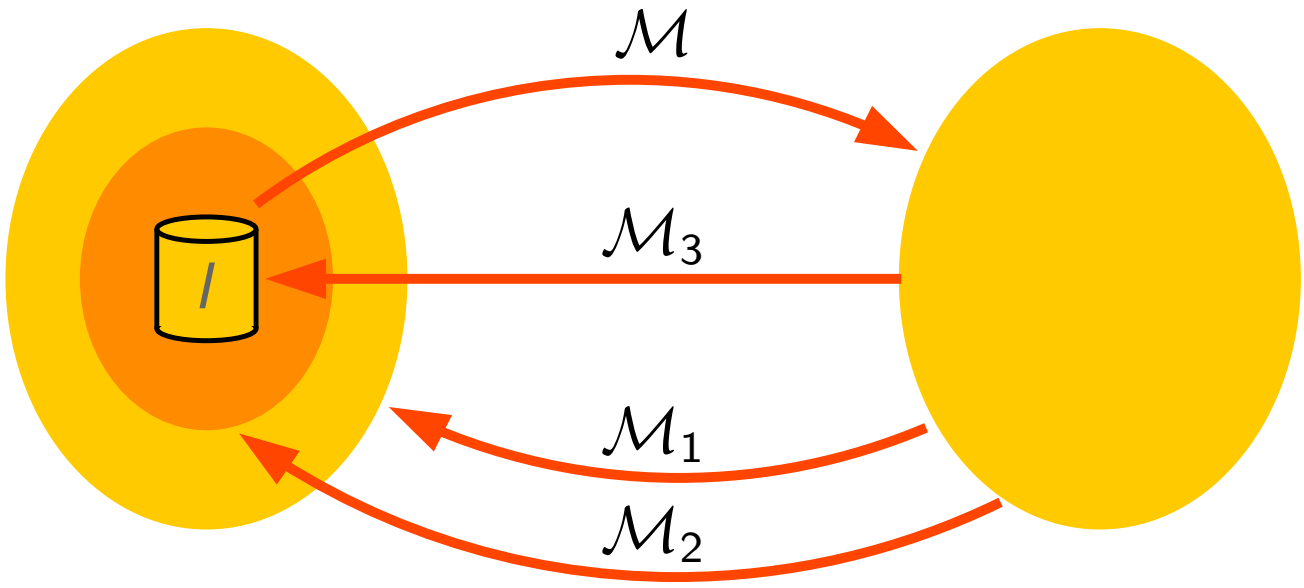


$\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$



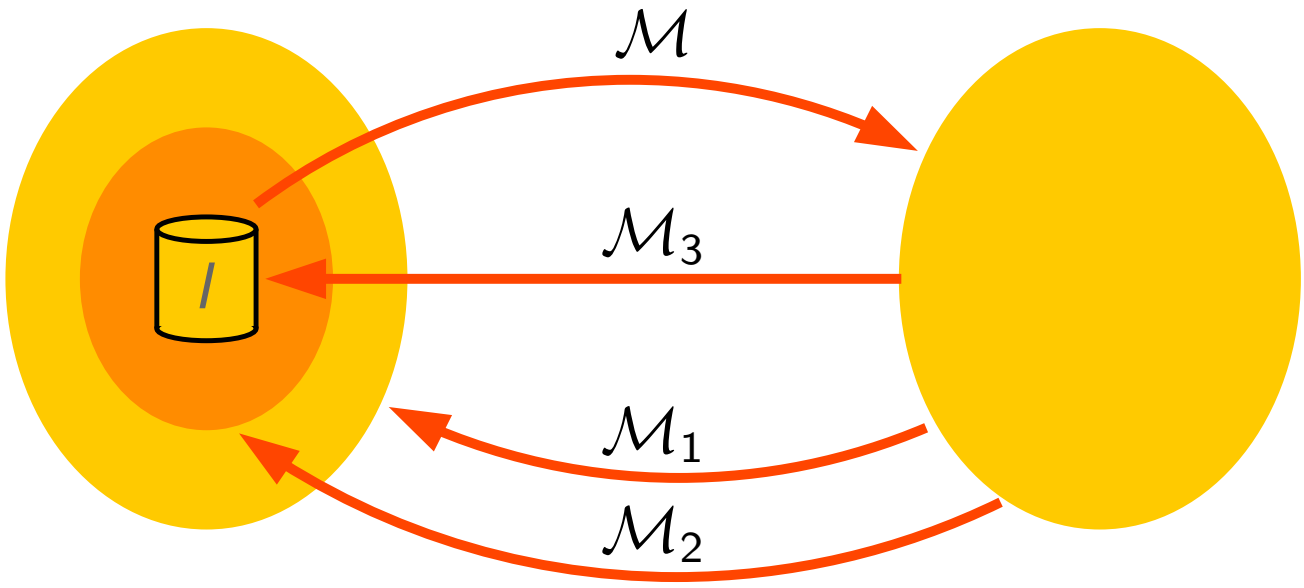
$\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$

$$\mathcal{M}'' \preceq \mathcal{M}'$$



$\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$

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



$\mathcal{M}'$ is *at least as informative as* $\mathcal{M}''$ for $\mathcal{M}$

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

Definition

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

A necessary and sufficient condition for the existence of maximum recoveries

A necessary and sufficient condition for the existence of maximum recoveries

Definition

J is a *witness solution* for I under $\mathcal{M}$ if for every other instance I',

$$J \in \mathrm{Sol}_{\mathcal{M}}(I') \implies \mathrm{Sol}_{\mathcal{M}}(I) \subseteq \mathrm{Sol}_{\mathcal{M}}(I').$$

A necessary and sufficient condition for the existence of maximum recoveries

Definition

J is a *witness solution* for I under $\mathcal{M}$ if for every other instance I',

$$J \in \mathrm{Sol}_{\mathcal{M}}(I') \implies \mathrm{Sol}_{\mathcal{M}}(I) \subseteq \mathrm{Sol}_{\mathcal{M}}(I').$$

Theorem

$\mathcal{M}$ has a maximum recovery iff every source instance has a witness solution.

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

- $\mathbf{L}_1$**-to-**$\mathbf{L}_2$ dependency: $\varphi_{\mathbf{S}}(\bar{x}) \in \mathbf{L}_1$ and $\psi_{\mathbf{T}}(\bar{x}) \in \mathbf{L}_2$.

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

- $\mathbf{L}_1$**-to-**$\mathbf{L}_2$ dependency: $\varphi_{\mathbf{S}}(\bar{x}) \in \mathbf{L}_1$ and $\psi_{\mathbf{T}}(\bar{x}) \in \mathbf{L}_2$.

Example

$$\exists y\ R(x, y)\ \rightarrow\ \exists z\ P(x, z)$$

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

- $\mathbf{L}_1$**-to-**$\mathbf{L}_2$ dependency: $\varphi_{\mathbf{S}}(\bar{x}) \in \mathbf{L}_1$ and $\psi_{\mathbf{T}}(\bar{x}) \in \mathbf{L}_2$.

Example

$$\exists y\ R(x, y) \ \rightarrow\ \exists z\ P(x, z)$$

CQ-to-CQ

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

- $\mathbf{L}_1$**-to-**$\mathbf{L}_2$ dependency: $\varphi_{\mathbf{S}}(\bar{x}) \in \mathbf{L}_1$ and $\psi_{\mathbf{T}}(\bar{x}) \in \mathbf{L}_2$.

Example

$$R(x, y) \rightarrow \exists z\ P(x, z)$$

CQ-to-CQ

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

- $\mathbf{L}_1$**-to-**$\mathbf{L}_2$ dependency: $\varphi_{\mathbf{S}}(\bar{x}) \in \mathbf{L}_1$ and $\psi_{\mathbf{T}}(\bar{x}) \in \mathbf{L}_2$.

Example

$$R(x, y) \rightarrow \exists z\ P(x, z)$$

CQ-to-CQ = st-tgd

FO-to-CQ dependencies and specification of mappings

- **FO-to-CQ** dependencies:

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

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

- $\mathbf{L}_1$**-to-**$\mathbf{L}_2$ dependency: $\varphi_{\mathbf{S}}(\bar{x}) \in \mathbf{L}_1$ and $\psi_{\mathbf{T}}(\bar{x}) \in \mathbf{L}_2$.

Example

$$R(x, y) \rightarrow \exists z\ P(x, z)$$

CQ-to-CQ = st-tgd

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

$$J \text{ is a solution for } I \text{ under } \mathcal{M} \iff (I, J) \models \Sigma.$$

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Example

$$\mathcal{M}: \quad P(x,y) \wedge P(y,z) \quad \rightarrow \quad R(x,z) \wedge T(y)$$

- Has neither an inverse nor a quasi-inverse [FKPT07].

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Example

$$\mathcal{M}: \quad P(x,y) \wedge P(y,z) \quad \rightarrow \quad R(x,z) \wedge T(y)$$

- Has neither an inverse nor a quasi-inverse [FKPT07].

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Example

$$\mathcal{M}: \quad P(x,y) \wedge P(y,z) \quad \rightarrow \quad R(x,z) \wedge T(y)$$

$$\mathcal{M}': \quad R(x,z) \quad \rightarrow \quad \exists Y \; P(x,Y) \wedge P(Y,z)$$

- Has neither an inverse nor a quasi-inverse [FKPT07].

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Example

$$\mathcal{M}: \quad P(x,y) \wedge P(y,z) \quad \rightarrow \quad R(x,z) \wedge T(y)$$

$$\mathcal{M}': \quad R(x,z) \quad \rightarrow \quad \exists Y \; P(x,Y) \wedge P(Y,z)$$

- Has neither an inverse nor a quasi-inverse [FKPT07].

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Example

$$\mathcal{M}: \quad P(x,y) \wedge P(y,z) \quad \rightarrow \quad R(x,z) \wedge T(y)$$

$$\mathcal{M}': \quad \begin{array}{rcl} R(x,z) & \rightarrow & \exists Y \; P(x,Y) \wedge P(Y,z) \\ T(y) & \rightarrow & \exists X \exists Z \; P(X,y) \wedge P(y,Z) \end{array}$$

- Has neither an inverse nor a quasi-inverse [FKPT07].

Every **FO-to-CQ** mapping has a maximum recovery.

Theorem

Every mapping specified by **FO-to-CQ** *dependencies has a maximum recovery.*

Proof idea

Every universal solution [FKMP03] is a witness solution. □

Example

$$\mathcal{M}: \quad P(x,y) \wedge P(y,z) \quad \rightarrow \quad R(x,z) \wedge T(y)$$

$$\begin{array}{rccl} \mathcal{M}': & R(x,z) & \rightarrow & \exists Y \; P(x,Y) \wedge P(Y,z) \\ & T(y) & \rightarrow & \exists X \exists Z \; P(X,y) \wedge P(y,Z) \end{array}$$

- Has neither an inverse nor a quasi-inverse [FKPT07].
- $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$

Maximum recoveries
are not guaranteed to exist in general.

Proposition

There are mappings specified by

- **CQ-to-CQ$^{\neq}$**
- **CQ-to-UCQ**
- **CQ-to-CQ$^{\neg}$**

dependencies, that have no maximum recovery.

Maximum recoveries strictly generalize [Fag06]-inverses

Maximum recoveries strictly generalize [Fag06]-inverses

$\mathcal{M}$ is *closed-down on the left* iff:

if J is a solution for I_2 and $I_1 \subseteq I_2 \implies J$ is a solution for I_1.

[Fag06] defines an *inverse notion* focusing on these mappings

Maximum recoveries strictly generalize [Fag06]-inverses

$\mathcal{M}$ is *closed-down on the left* iff:

if J is a solution for I_2 and $I_1 \subseteq I_2 \implies J$ is a solution for I_1.

[Fag06] defines an *inverse notion* focusing on these mappings

Theorem

If $\mathcal{M}$ is closed-down on the left and invertible, then

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

Maximum recoveries strictly generalize [Fag06]-inverses

$\mathcal{M}$ is *closed-down on the left* iff:

if J is a solution for I_2 and $I_1 \subseteq I_2 \implies J$ is a solution for I_1.

[Fag06] defines an *inverse notion* focusing on these mappings

Theorem

If $\mathcal{M}$ is closed-down on the left and invertible, then

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

Example (not closed-down on the left)

$$\mathcal{M}\colon\ P(x) \leftrightarrow R(x)$$

Maximum recoveries strictly generalize [Fag06]-inverses

$\mathcal{M}$ is *closed-down on the left* iff:

if J is a solution for I_2 and $I_1 \subseteq I_2 \implies J$ is a solution for I_1.

[Fag06] defines an *inverse notion* focusing on these mappings

Theorem

If $\mathcal{M}$ is closed-down on the left and invertible, then

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

Example (not closed-down on the left)

$$\mathcal{M}\colon\ P(x) \leftrightarrow R(x)$$

$$\mathcal{M}'\colon\ R(x) \leftrightarrow P(x).$$

Maximum recoveries strictly generalize [Fag06]-inverses

$\mathcal{M}$ is *closed-down on the left* iff:

if J is a solution for I_2 and $I_1 \subseteq I_2 \implies J$ is a solution for I_1.

[Fag06] defines an *inverse notion* focusing on these mappings

Theorem

If $\mathcal{M}$ is closed-down on the left and invertible, then

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

Example (not closed-down on the left)

$$\mathcal{M}\colon\ P(x) \leftrightarrow R(x)$$

$$\mathcal{M}'\colon\ R(x) \leftrightarrow P(x).$$

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

Maximum recoveries strictly generalize [Fag06]-inverses

$\mathcal{M}$ is *closed-down on the left* iff:

if J is a solution for I_2 and $I_1 \subseteq I_2 \implies J$ is a solution for I_1.

[Fag06] defines an *inverse notion* focusing on these mappings

Theorem

If $\mathcal{M}$ is closed-down on the left and invertible, then

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

Example (not closed-down on the left)

$$\mathcal{M}\colon\ P(x) \leftrightarrow R(x)$$

$$\mathcal{M}'\colon\ R(x) \leftrightarrow P(x).$$

$\mathcal{M}'$ is a maximum recovery *but not* a [Fag06]-inverse of $\mathcal{M}$.

Maximum recoveries help in the study of invertibility.

Maximum recoveries help in the study of invertibility.

Theorem in [FKPT07]:
subset property characterizes invertibility for the **CQ-to-CQ** case.

Maximum recoveries help in the study of invertibility.

Theorem in [FKPT07]:
subset property characterizes invertibility for the **CQ-to-CQ** case.

Proposition

The subset property fails to characterize invertibility *for* **CQ-to-UCQ** *and for* **CQ-to-CQ**$^{\neq}$ *dependencies.*

Maximum recoveries help in the study of invertibility.

Theorem in [FKPT07]:
subset property characterizes invertibility for the **CQ-to-CQ** case.

Proposition

The subset property fails to characterize invertibility *for* **CQ-to-UCQ** *and for* **CQ-to-CQ**$^{\neq}$ *dependencies.*

Theorem

$\mathcal{M}$ *closed-down on the left is invertible if and only if*

- *satisfies the subset property, and*
- has a maximum recovery.

Maximum recoveries help in the study of invertibility.

Theorem in [FKPT07]:
subset property characterizes invertibility for the **CQ-to-CQ** case.

Proposition

The subset property fails to characterize invertibility *for* **CQ-to-UCQ** *and for* **CQ-to-CQ**$^{\neq}$ *dependencies.*

Theorem

$\mathcal{M}$ *closed-down on the left is invertible if and only if*

- *satisfies the subset property, and*
- has a maximum recovery.

In the paper: similar results for quasi-inverses.

Outline

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_1 : \begin{array}{lcl} \alpha(x) & \rightarrow & P(x) \\ \beta(y) & \rightarrow & P(y) \\ \gamma(z) & \rightarrow & R(z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_1 : \quad \begin{array}{lcl} \alpha(x) & \rightarrow & P(x) \\ \beta(y) & \rightarrow & P(y) \\ \gamma(z) & \rightarrow & R(z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_1: \quad \begin{array}{lcl} \alpha(x) & \rightarrow & P(x) \\ \beta(y) & \rightarrow & P(y) \\ \gamma(z) & \rightarrow & R(z) \end{array} \qquad \alpha(x) \vee \beta(x) \quad \rightarrow \quad P(x)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_1: \quad \begin{array}{lcl} \alpha(x) & \rightarrow & P(x) \\ \beta(y) & \rightarrow & P(y) \\ \gamma(z) & \rightarrow & R(z) \end{array} \qquad \begin{array}{lcl} \alpha(x) \vee \beta(x) & \rightarrow & P(x) \\ \gamma(z) & \rightarrow & R(z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_1: \quad \begin{array}{lcl} \alpha(x) & \rightarrow & P(x) \\ \beta(y) & \rightarrow & P(y) \\ \gamma(z) & \rightarrow & R(z) \end{array} \quad \equiv \quad \begin{array}{lcl} \alpha(x) \vee \beta(x) & \rightarrow & P(x) \\ \gamma(z) & \rightarrow & R(z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_1: \begin{array}{lcl} \alpha(x) & \rightarrow & P(x) \\ \beta(y) & \rightarrow & P(y) \\ \gamma(z) & \rightarrow & R(z) \end{array} \quad \equiv \quad \begin{array}{lcl} \alpha(x) \vee \beta(x) & \rightarrow & P(x) \\ \gamma(z) & \rightarrow & R(z) \end{array}$$

$$\mathcal{M}_1': \begin{array}{lcl} P(x) & \rightarrow & \alpha(x) \vee \beta(x) \\ R(z) & \rightarrow & \gamma(z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{ll} \alpha(x,y) & \rightarrow \; T(x,y) \\ \beta(z) & \rightarrow \; T(z,z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array}$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array} \qquad (\beta(x) \wedge x = y)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array}$$

$$(\beta(x) \wedge x = y) \rightarrow T(x,y)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{ccc} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array}$$

$$(\beta(x) \wedge x = y) \rightarrow T(x,y)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array} \qquad \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array} \quad \equiv \quad \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array} \quad \equiv \quad \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

$$\mathcal{M}_2' : \quad T(x,y) \quad \rightarrow \quad (\beta(x) \wedge x = y) \vee \alpha(x,y)$$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full FO-to-CQ)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array} \quad \equiv \quad \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

$$\mathcal{M}_2' : \quad T(x,y) \quad \rightarrow \quad (\beta(x) \wedge x = y) \vee \alpha(x,y)$$

Algorithm (full FO-to-CQ)

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full **FO-to-CQ**)

$$\mathcal{M}_2 : \begin{array}{ll} \alpha(x,y) & \rightarrow T(x,y) \\ \beta(z) & \rightarrow T(z,z) \end{array} \equiv \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

$$\mathcal{M}_2' : \quad T(x,y) \quad \rightarrow \quad (\beta(x) \wedge x = y) \vee \alpha(x,y)$$

Algorithm (full **FO-to-CQ**)

1. For every dependency $\varphi(\bar{x}) \rightarrow R(\bar{x})$

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full FO-to-CQ)

$$\mathcal{M}_2 : \begin{array}{ll} \alpha(x,y) & \rightarrow T(x,y) \\ \beta(z) & \rightarrow T(z,z) \end{array} \equiv \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

$$\mathcal{M}_2' : \quad T(x,y) \quad \rightarrow \quad (\beta(x) \wedge x = y) \vee \alpha(x,y)$$

Algorithm (full FO-to-CQ)

1. For every dependency $\varphi(\bar{x}) \rightarrow R(\bar{x})$
   - search for dependencies with R in the conclusion.

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full FO-to-CQ)

$$\mathcal{M}_2 : \begin{array}{ll} \alpha(x,y) & \rightarrow T(x,y) \\ \beta(z) & \rightarrow T(z,z) \end{array} \equiv \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

$$\mathcal{M}_2' : \quad T(x,y) \quad \rightarrow \quad (\beta(x) \wedge x = y) \vee \alpha(x,y)$$

Algorithm (full FO-to-CQ)

1. For every dependency $\varphi(\bar{x}) \rightarrow R(\bar{x})$
   - search for dependencies with R in the conclusion.
   - compose corresponding premises (using $\vee$ and $=$).

Algorithm idea: *explicit the relationships* in the mapping, then *reverse the arrows*.

Example (full FO-to-CQ)

$$\mathcal{M}_2 : \begin{array}{rcl} \alpha(x,y) & \rightarrow & T(x,y) \\ \beta(z) & \rightarrow & T(z,z) \end{array} \equiv \alpha(x,y) \vee (\beta(x) \wedge x = y) \rightarrow T(x,y)$$

$$\mathcal{M}_2' : \quad T(x,y) \quad \rightarrow \quad (\beta(x) \wedge x = y) \vee \alpha(x,y)$$

Algorithm (full FO-to-CQ)

1. For every dependency $\varphi(\bar{x}) \rightarrow R(\bar{x})$
   - search for dependencies with R in the conclusion.
   - compose corresponding premises (using $\vee$ and $=$).
2. Reverse the arrows.

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \longrightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \longrightarrow & P(u,v) \\ \gamma(s,t) & \longrightarrow & R(s,t) \end{array}$$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(u,v) \wedge \gamma(s,t) \quad \rightarrow \quad P(u,v) \wedge R(s,t)$$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(u,v) \wedge \gamma(s,t) \quad \rightarrow \quad P(u,v) \wedge R(s,t)$$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(u,v) \wedge \gamma(s,t) \quad \rightarrow \quad \exists z\ P(u,z) \wedge R(z,t)$$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(u,v) \wedge \gamma(s,t) \wedge v = s \quad \rightarrow \quad \exists z\ P(u,z) \wedge R(z,t)$$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(u,v) \wedge \gamma(s,t) \wedge v = s \quad \rightarrow \quad \exists z\ P(u,z) \wedge R(z,t)$$

Existential replacement

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\ P(x,z) \wedge R(z,y)$$

Existential replacement

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\ P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\ P(x,z) \wedge R(z,y)$$

Existential replacement

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\, P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\, P(x,z) \wedge R(z,y)$$

Existential replacement

Algorithm (FO-to-CQ)

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\, P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\, P(x,z) \wedge R(z,y)$$

Existential replacement

Algorithm (FO-to-CQ)

1. For every dependency $\sigma : \varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\, P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\, P(x,z) \wedge R(z,y)$$

Existential replacement

Algorithm (FO-to-CQ)

1. For every dependency $\sigma : \varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$
   - search for combinations of conclusions that can be made equal to $\exists \bar{y} \psi(\bar{x}, \bar{y})$ by *existential replacements*.

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\, P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\, P(x,z) \wedge R(z,y)$$

Existential replacement

Algorithm (FO-to-CQ)

1. For every dependency $\sigma : \varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$
   - search for combinations of conclusions that can be made equal to $\exists \bar{y} \psi(\bar{x}, \bar{y})$ by *existential replacements*.
   - compose corresponding premises (using $\vee$ and $=$).

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\, P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\, P(x,z) \wedge R(z,y)$$

Existential replacement

Algorithm (FO-to-CQ)

1. For every dependency $\sigma : \varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$
   - search for combinations of conclusions that can be made equal to $\exists \bar{y} \psi(\bar{x}, \bar{y})$ by *existential replacements*.
   - compose corresponding premises (using $\vee$ and $=$).
2. Reverse the arrows

Expliciting in the non-full case is more complicated.

Example

$$\begin{array}{rcl} (\ldots) \vee \alpha(x,y) & \rightarrow & \exists z\, P(x,z) \wedge R(z,y) \\ \beta(u,v) & \rightarrow & P(u,v) \\ \gamma(s,t) & \rightarrow & R(s,t) \end{array}$$

$$\beta(x,v) \wedge \gamma(s,y) \wedge v = s \quad \rightarrow \quad \exists z\, P(x,z) \wedge R(z,y)$$

Existential replacement

Algorithm (FO-to-CQ)

1. For every dependency $\sigma : \varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$
   - search for combinations of conclusions that can be made equal to $\exists \bar{y} \psi(\bar{x}, \bar{y})$ by *existential replacements*.
   - compose corresponding premises (using $\vee$ and $=$).
2. Reverse the arrows forcing *constants* in the source (**Const**$(\cdot)$).

Algorithm: quadratic in the full case, exponential in general

	full **FO-to-CQ**	**FO-to-CQ**
Output:		
Size:		
Time:		

Algorithm: quadratic in the full case, exponential in general

	full **FO-to-CQ**	**FO-to-CQ**
Output:	**CQ-to-FO**	
Size:	quadratic	
Time:	quadratic	

Algorithm: quadratic in the full case, exponential in general

	full **FO-to-CQ**	**FO-to-CQ**
Output:	**CQ-to-FO**	**CQ$^{\mathbf{Const}(\cdot)}$-to-FO**
Size:	quadratic	exponential
Time:	quadratic	exponential

Algorithm: quadratic in the full case, exponential in general

	full **FO-to-CQ**	**FO-to-CQ**
Output:	**CQ-to-FO**	**CQ$^{\mathbf{Const}(\cdot)}$-to-FO**
Size:	quadratic	exponential
Time:	quadratic	exponential

Some *highlights*:

- Obtains small inverses and quasi-inverses in the full case (exponential in [FKPT07]).

Algorithm: quadratic in the full case, exponential in general

	full **FO-to-CQ**	**FO-to-CQ**
Output:	**CQ-to-FO**	**CQ$^{\mathbf{Const}(\cdot)}$-to-FO**
Size:	quadratic	exponential
Time:	quadratic	exponential

Some *highlights*:

- Obtains small inverses and quasi-inverses in the full case (exponential in [FKPT07]).
- Obtains inverses and quasi-inverses in a more general setting (only for **CQ-to-CQ** in [FKPT07]).

Algorithm: quadratic in the full case, exponential in general

	full **FO-to-CQ**	**FO-to-CQ**
Output:	**CQ-to-FO**	$\mathbf{CQ^{Const(\cdot)}}$**-to-FO**
Size:	quadratic	exponential
Time:	quadratic	exponential

Some *highlights*:

- Obtains small inverses and quasi-inverses in the full case (exponential in [FKPT07]).
- Obtains inverses and quasi-inverses in a more general setting (only for **CQ-to-CQ** in [FKPT07]).

In the paper: a formal justification of $\mathbf{CQ^{Const(\cdot)}}$**-to-FO** as the language needed to specify maximum recoveries.

Complexity results

Complexity results

Theorem

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* recovery *of* $\mathcal{M}$*?*

Complexity results

Theorem

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* recovery *of* $\mathcal{M}$*?*

coNP-complete *if full* $\mathcal{M}$ *and* $\mathcal{M}'$

Complexity results

Theorem

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* recovery *of* $\mathcal{M}$*?*

$$\begin{array}{rl} \textit{coNP-complete} & \textit{if full } \mathcal{M} \textit{ and } \mathcal{M}' \\ \Pi_2^P\textit{-complete} & \textit{if full } \mathcal{M} \end{array}$$

Complexity results

Theorem

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* recovery *of* $\mathcal{M}$*?*

coNP-complete	*if full* $\mathcal{M}$ *and* $\mathcal{M}'$
Π_2^P*-complete*	*if full* $\mathcal{M}$
undecidable	*in general*

Complexity results

Theorem

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* recovery *of* $\mathcal{M}$*?*

coNP-complete	*if full* $\mathcal{M}$ *and* $\mathcal{M}'$
Π_2^P*-complete*	*if full* $\mathcal{M}$
undecidable	*in general*

Corollary

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* maximum recovery *of* $\mathcal{M}$*?*

undecidable

Complexity results

Theorem

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* recovery *of* $\mathcal{M}$*?*

coNP-complete	*if full* $\mathcal{M}$ *and* $\mathcal{M}'$
Π_2^P*-complete*	*if full* $\mathcal{M}$
undecidable	*in general*

Corollary

Input: $\mathcal{M}$ *and* $\mathcal{M}'$ **CQ-to-CQ** *dependencies*
Question: *Is* $\mathcal{M}'$ *a* maximum recovery *of* $\mathcal{M}$*?*

undecidable

Also undecidable *for inverses and quasi-inverses.*

Outline

Maximum recovery: alternative notion on how to *reverse* schema mappings.

Maximum recovery: alternative notion on how to *reverse* schema mappings.

Conceptually, a new notion on how to reverse mappings

- simple, following a *best effort* approach
- general, not tailored to any specific language or data model

Maximum recovery: alternative notion on how to *reverse* schema mappings.

Conceptually, a new notion on how to reverse mappings

- simple, following a *best effort* approach
- general, not tailored to any specific language or data model

Technically:

- characterization of the existence of maximum recoveries
- positive result for the most common type of mappings
- algorithm for a large class of mappings
- complexity results
- new and general results for inverses and quasi-inverses

The Recovery of a Schema Mapping: Bringing Exchanged Data Back

Marcelo Arenas Jorge Pérez Cristian Riveros

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Outline