Lecture 7: Query Answering by Rewriting, Mapping Management
28 May 2020

Informationssysteme CS4130
(Summer 2020)
Query Answering
Remember: Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$

- Semantics of query answering specified as certain answer semantics

Definition

The certain answers of query $Q$ over $\tau$ for given instance $\mathcal{G}$ is defined as

$$\text{cert}_\mathcal{M}(Q, \mathcal{G}) = \bigcap \{ \text{cert}(Q, \mathcal{I}) \mid \mathcal{I} \in \text{SOL}_\mathcal{M}(\mathcal{G}) \}$$
Remember: Certain Answers

- Given mapping \( \mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau) \)

- Semantics of query answering specified as certain answer semantics

**Definition**

The *certain answers* of query \( Q \) over \( \tau \) for given instance \( \mathcal{G} \) is defined as

\[
\text{cert}_\mathcal{M}(Q, \mathcal{G}) = \bigcap \{ \text{cert}(Q, \Xi) \mid \Xi \in \text{SOL}_\mathcal{M}(\mathcal{G}) \}
\]

- We saw: In many cases it is not necessary to compute all solutions to get certain answers \( \implies \) universal solutions
Remember: Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$

- Semantics of query answering specified as certain answer semantics

Definition

The certain answers of query $Q$ over $\tau$ for given instance $\mathcal{G}$ is defined as

$$cert_{\mathcal{M}}(Q, \mathcal{G}) = \bigcap \{ cert(Q, \mathcal{T}) \mid \mathcal{T} \in SOL_{\mathcal{M}}(\mathcal{G}) \}$$

- We saw: In many cases it is not necessary to compute all solutions to get certain answers $\implies$ universal solutions
- But as universal solution $\mathcal{T}$ (usually) is an incomplete DB, we would have to consider all completions (requires: $cert(Q, \mathcal{T})$)
Remember: Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$

- Semantics of query answering specified as certain answer semantics

**Definition**

The certain answers of query $Q$ over $\tau$ for given instance $\mathcal{G}$ is defined as

$$\text{cert}_\mathcal{M}(Q, \mathcal{G}) = \bigcap\{ \text{cert}(Q, \mathcal{T}) \mid \mathcal{T} \in \text{SOL}\mathcal{M}(\mathcal{G}) \}$$

- We saw: In many cases it is not necessary to compute all solutions to get certain answers $\implies$ universal solutions
- But as universal solution $\mathcal{T}$ (usually) is an incomplete DB, we would have to consider all completions (requires: $\text{cert}(Q, \mathcal{T})$)
- Sometimes this is not required $\implies$ Query rewriting
Definition (Naive evaluation strategy for general DBs)

For an arbitrary general DB $\mathcal{G}$ the set of answers following a naive evaluation strategy, for short $Q_{\text{naive}}(\mathcal{G})$, is calculated as follows:

- Treat marked NULLS in $\mathcal{G}$ as constants (i.e. $\bot = \bot$ is true but not $\bot = c$ and not $\bot = \bot'$)
- Calculate $Q(\mathcal{G})$ under this perspective (treating $\mathcal{G}$ as ordinary complete DB)
- and then eliminate all tuples from $Q(\mathcal{G})$ containing a NULL
Certain Answers Naively

Theorem

For UCQs $Q$:

$$\text{cert}(\mathcal{G}, Q) = Q_{\text{naive}}(\mathcal{G})$$

Proof sketch:

- For every $\mathcal{G}' \in \text{Rep}(\mathcal{G})$ there is $\mathcal{G} \xrightarrow{\text{hom}} \mathcal{G}'$
- As homomorphisms preserve answers of CQs:
  $$Q_{\text{naive}}(\mathcal{G}) = \text{NULL-free tuples in } Q(\mathcal{G}) \subseteq \bigcap_{\mathcal{G}' \in \text{Rep}(\mathcal{G})} Q(\mathcal{G}')$$
- $$Q_{\text{naive}}(\mathcal{G}) \supseteq \bigcap_{\mathcal{G}' \in \text{Rep}(\mathcal{G})} Q(\mathcal{G}')$$
  because $\mathcal{G}$ can be considered as its own completion (when treating NULLs consistently as constants).

Definition (Naive Evaluation Strategy for DEs)

\[ \text{cert}_M(\mathcal{G}, Q) = Q_{naive}(\mathcal{I}) \]

where \( \mathcal{I} \) is a universal solution for \( M \) and \( \mathcal{G} \).
Use of naive strategy for DE

Definition (Naive Evaluation Strategy for DEs)

\[ cert_M (\mathcal{G}, Q) = Q_{naive}(\mathcal{I}) \]

where \( \mathcal{I} \) is a universal solution for \( M \) and \( \mathcal{G} \).

- This strategy works also for Datalog programs as constraints for the target schema \( \tau \)
  - Reason: Datalog programs are preserved under homomorphisms
  - Even if one adds inequalities, naive evaluation works
  - Hence certain answering is here in PTime
Rewritability

- Naive evaluation is a form of rewriting
- Again: Fundamental method that re-appears in different areas of CS
- Rewrite a query w.r.t. a given KB into a new query that “contains” the knowledge of KB

- Challenges
  - Preserve the semantics in the rewriting process: ensure correctness (easy) and completeness (difficult)
  - The language of the output query is constraint to a “simple language” (so rewritability not always guaranteed)
Rewritability for DE

**Definition (FOL Rewritability)**

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ be a mapping and $Q$ be a query over $\tau$.

Then $Q$ is said to be FOL-rewritable over canonical universal solutions $(\mathcal{S})$ under $\mathcal{M}$ iff there is a FOL query $Q_{rew}$ over $\tau^C$ s.t.

$$\text{cert}_{\mathcal{M}}(Q, \mathcal{S}) = Q_{rew}(\mathcal{S})$$

- Here $\tau^C = \tau \cup \{C\}$ where unary predicate $C$ depicts all constants (not NULLs) in targets
- $C$ works like a type predicate
Rewritability for DE

**Definition (FOL Rewritability)**

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ be a mapping and $Q$ be a query over $\tau$.

Then $Q$ is said to be FOL-rewritable over canonical universal solutions ($\mathcal{T}$) under $\mathcal{M}$ iff there is a FOL query $Q_{rew}$ over $\tau^C$ s.t.

$$\text{cert}_\mathcal{M}(Q, \mathcal{S}) = Q_{rew}(\mathcal{T})$$

Note: One must find one rewriting for any given pair of source $\mathcal{S}$ and universal solution $\mathcal{T}$

- The known component is the mapping $\mathcal{M}$
- The unknown components are all pairs ($\mathcal{S}, \mathcal{T}$)
Rewritability for DE

**Definition (FOL Rewritability)**

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ be a mapping and $Q$ be a query over $\tau$.

Then $Q$ is said to be **FOL-rewritable over canonical universal solutions under $\mathcal{M}$** iff there is a FOL query $Q_{\text{rew}}$ over $\tau^C$ such that

$$\text{cert}_\mathcal{M}(Q, \mathcal{S}) = Q_{\text{rew}}(\mathcal{I})$$

If, in the definition, one talks about cores $\mathcal{I}$ instead of universal solutions then $Q$ is said to be **FOL-rewritable over cores**

**Theorem**

*For mappings without target dependencies:*

**FOL-rewrit. over core** $\implies$ **FOL-rewrit. over universal solution, but not vice versa.**
Rewritability for DE

Definition (FOL-Rewritability)

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ be a mapping and $Q$ be a query over $\tau$. Then $Q$ is said to be FOL-rewritable over canonical universal solutions under $\mathcal{M}$ iff there is a FOL query $Q_{\text{rew}}$ over $\tau^C$ such that

$$\text{cert}_{\mathcal{M}}(Q, \mathcal{S}) = Q_{\text{rew}}(\mathcal{I})$$

Example

- $Q(\overline{x})$: a conjunctive query
- $Q_{\text{rew}}: Q(\overline{x}) \land C(x_1) \land \cdots \land C(x_n)$
  This is actually the syntactic form of $Q_{\text{naive}}$
- The rewriting is even independent of $\mathcal{M}$
- So: (U)CQs are rewritable for any mapping
Adding Negations to Query Language

- Negations in query languages lead to loss of naive rewriting technique
- Even if one allows negation only within inequalities

**Definition (Conjunctive Queries with inequalities \( CQ\neq \))**

A conjunctive query with inequalities is a query of the form

\[
Q(\vec{x}) = \exists \vec{y} \left( \alpha_1(\vec{x}_1, \vec{y}_1) \land \cdots \land \alpha_n(\vec{x}_n, \vec{y}_n) \right)
\]

where \( \alpha_i \) is either an atomic relational formula or an inequality \( z_i \neq z_j \).
Example (No Naive Evaluation Possible)

<table>
<thead>
<tr>
<th>Source DB</th>
<th>Target DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight (src, dest, airl, dep)</td>
<td>Routes(fno, src, dest)</td>
</tr>
<tr>
<td>paris</td>
<td>airFr</td>
</tr>
<tr>
<td>sant.</td>
<td>lan</td>
</tr>
</tbody>
</table>

▶ Dependencies $M_{\sigma_T}$

$$\text{Flight}(src, dest, airl, dep) \rightarrow \exists fno \exists arr(\text{Routes}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))$$
Example (No Naive Evaluation Possible)

Source DB
Flight ( src, dest, airl, dep )
  paris sant. airFr 2320
  paris sant. lan 2200

Target DB
Routes( fno, src, dest )
Info( fno, dep, arr, airl )

 Dependencies \( \mathcal{M}_{\sigma \tau} \)

\[
\begin{align*}
\text{Flight}(src, dest, airl, dep) \rightarrow \\
\exists \ fno \ \exists \ arr(\text{Routes}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))
\end{align*}
\]

Any universal solution \( \mathcal{Z}' \) contains as sub-instance universal \( \tau\)-solution

\[
\mathcal{Z} = \{ \text{Routes}(\bot_1, \text{paris}, \text{sant}), \text{Info}(\bot_1, 2320, \bot_2, \text{airFr}), \\
\quad \text{Routes}(\bot_3, \text{paris}, \text{sant}), \text{Info}(\bot_3, 2320, \bot_4, \text{lan}) \}
\]
Example (No Naive Evaluation Possible)

<table>
<thead>
<tr>
<th>Source DB</th>
<th>Target DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight (</td>
<td></td>
</tr>
<tr>
<td>src, dest, airl, dep )</td>
<td></td>
</tr>
<tr>
<td>Paris sant. airFr 2320</td>
<td></td>
</tr>
<tr>
<td>Paris sant. lan 2200</td>
<td></td>
</tr>
<tr>
<td>Routes( fno, src, dest )</td>
<td></td>
</tr>
<tr>
<td>Info( fno, dep, arr, airl )</td>
<td></td>
</tr>
</tbody>
</table>

▶ Dependencies $M_{\sigma_T}$

$$
\text{Flight}(src, \text{dest}, airl, \text{dep}) \rightarrow \\
\exists fno \exists arr (\text{Routes}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))
$$

▶ Any universal solution $\mathcal{\exists'}$ contains as sub-instance universal $\tau$-solution

$$
\mathcal{\exists} = \{ \text{Routes}(\perp_1, \text{paris, sant}), \ \text{Info}(\perp_1, 2320, \perp_2, \text{airFr}), \\
\text{Routes}(\perp_3, \text{paris, sant}), \ \text{Info}(\perp_3, 2320, \perp_4, \text{lan}) \}
$$

▶ Query $Q(x, z) = \exists y \exists y' (\text{Routes}(y, x, z) \land \text{Routes}(y', x, z) \land y \neq y')$
### Example (No Naive Evaluation Possible)

<table>
<thead>
<tr>
<th>Source DB</th>
<th>Target DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight ( src, dest, airl, dep )</td>
<td>Routes( fno, src, dest )</td>
</tr>
<tr>
<td>paris sant. airFr 2320</td>
<td>Info( fno, dep, arr, airl )</td>
</tr>
<tr>
<td>paris sant. lan 2200</td>
<td></td>
</tr>
</tbody>
</table>

** Dependencies \( M_{\sigma\tau} \)**

\[
\text{Flight}(src, dest, airl, dep) \longrightarrow \\
\exists fno \exists arr(\text{Routes}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))
\]

** Any universal solution \( \mathcal{U}' \) contains as sub-instance universal \( \tau\text{-solution} \)**

\[
\mathcal{U} = \{ \text{Routes}(\bot_1, \text{paris}, \text{sant}), \text{Info}(\bot_1, 2320, \bot_2, \text{airFr}), \text{Routes}(\bot_3, \text{paris}, \text{sant}), \text{Info}(\bot_3, 2320, \bot_4, \text{lan}) \}
\]

** Query \( Q(x, z) = \exists y \exists y' (\text{Routes}(y, x, z) \land \text{Routes}(y', x, z) \land y \neq y') \)**

** \( Q_{\text{naive}}(\mathcal{U}') = \{(\text{paris, sant})\} \) (for any universal solution \( \mathcal{U}' \))**
Example (No Naive Evaluation Possible)

Source DB

Flight ( src, dest, airl, dep )

paris  sant.  airFr  2320
paris  sant.  lan   2200

Target DB

Routes( fno, src, dest )

Info( fno, dep, arr, airl )

▶ Dependencies $M_{\sigma\tau}$

$\text{Flight}(src, dest, airl, dep) \rightarrow \exists fno \exists arr(\text{Routes}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))$

▶ Any universal solution $\mathcal{T}'$ contains as sub-instance universal $\tau$-solution

$\mathcal{T} = \{ \text{Routes}(\bot_1, paris, sant), \text{Info}(\bot_1, 2320, \bot_2, airFr), \text{Routes}(\bot_3, paris, sant), \text{Info}(\bot_3, 2320, \bot_4, lan) \}$

▶ Query $Q(x, z) = \exists y \exists y'(\text{Routes}(y, x, z) \land \text{Routes}(y', x, z) \land y \neq y')$

$Q_{\text{naive}}(\mathcal{T}') = \{(paris, sant)\}$ (for any universal solution $\mathcal{T}'$)

▶ But: $\text{cert}_M(Q(x, z), \mathcal{G}) = \emptyset$ because there is a solution

$\mathcal{T}'' = \{ \text{Routes}(\bot_1, paris, sant), \text{Info}(\bot_1, 2320, \bot_2, airFr), \text{Info}(\bot_1, 2320, \bot_2, lan) \}$
$CQ \neq$ is in $\text{coNP}$

- In case of $CQ \neq$ one cannot even find a tractable means to answer them w.r.t. certain answer semantics

**Theorem**

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ be a mapping where $M_{\tau}$ is the union of egds and weakly acyclic tgds, and let $Q$ be a $\text{UCQ} \neq$ query. Then:

$\text{CERTAIN}_\mathcal{M}(Q)$ is in $\text{coNP}$
Non-rewritability

- Generally it is not possible to decide whether rewritability holds

**Theorem**

*For mappings without target constraints one can not decide whether a given FOL query is rewritable over the canonical solutions (over the core).*

- Showing Non-FOL-rewritability can be done with locality tools
- Actually: One uses (adapted) Hanf-locality
Not Covered in our DE Lectures

- Different semantics for query answering
  - Combinations of open-world (certain answers) and closed-word semantics

- DE for non-relational DBs
  - e.g., DE for semi-structured data (XML)
  - requires techniques other than that for relational DE
Not Covered in our DE Lectures

- Different semantics for query answering
  - Combinations of open-world (certain answers) and closed-word semantics

- DE for non-relational DBs
  - e.g., DE for semi-structured data (XML)
  - requires techniques other than that for relational DE

- Rest of this lecture: mapping management
  - How to maintain mappings w.r.t. consistency (only a few remarks today)
  - How to compose mappings
  - How to invert mappings: Get back source DB from target DB
Motivation Mapping Management
Consistency of Mappings

- So far: Considered existence of $\tau$-solutions given $\sigma$-instance in mapping $M$
- Now: Given only $M$
  - consistency/local consistency of $M$: Is there a $\sigma$-instance s.t. there is a $\tau$-solution
  - Absolute consistency/Global consistency: Is there for each $\sigma$-instance a $\tau$-solution?
Mapping Evolution

- Mappings may change due to schema evolution
  - Target schema changes: need **composition of mappings**
  - Source schema changes: need **inverse of mappings**
Mapping Evolution

- Mappings may change due to schema evolution
  - Target schema changes: need composition of mappings
  - Source schema changes: need inverse of mappings
  - Can think of other operations (merge of mappings ... )
Composition for Target Schema Change

source schema $\sigma$ $\xrightarrow{M_{\sigma\tau}}$ target schema $\tau$

$\sigma$ DB $\xrightarrow{\text{Exchange}}$ $\tau$ DB

$\tau$ query

materialized
Composition for Target Schema Change

source schema $\sigma$ $\rightarrow$ mapping rules $M_{\sigma\tau}$ $\rightarrow$ target schema $\tau$ $\rightarrow$ mapping rules $M_{\tau\tau'}$ $\rightarrow$ target schema $\tau'$

$\sigma$ DB $\rightarrow$ Exchange $\rightarrow$ $\tau$ DB

materialized
Composition for Target Schema Change

source schema $\sigma$

mapping rules $M_{\sigma\tau}$

target schema $\tau$

mapping rules $M_{\tau\tau'}$

query

$\tau'$

exchange

$\sigma$ DB

materialized

$\tau$ DB

materialized

$\tau'$ DB
Composition for Target Schema Change

Composed mapping $M_{\sigma \tau} \circ M_{\tau \tau'}$

- Source schema $\sigma$
- Target schema $\tau$
- Target schema $\tau'$

- Mapping rules $M_{\sigma \tau}$
- Mapping rules $M_{\tau \tau'}$

- Query $\tau'$

- Database $\sigma$ DB
- Database $\tau'$ DB

Exchange

Materialized
Example (DE in Flight Domain)

<table>
<thead>
<tr>
<th>Source schema $\sigma$</th>
<th>Target schema $\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo( city, coun, pop )</td>
<td>Route( fno, src, dest )</td>
</tr>
<tr>
<td>Flight( src, dest, airl, dep )</td>
<td>Info( fno, dep, arr, airl )</td>
</tr>
<tr>
<td></td>
<td>Serves( airl, city, coun, phone )</td>
</tr>
</tbody>
</table>

Mapping rules $M$

1. $\sigma(\text{Flight}(\text{src}, \text{dest}, \text{airl}, \text{dep})) \rightarrow \exists \text{fno} \exists \text{arr} (\tau(\text{Route}(\text{fno}, \text{src}, \text{dest})) \land \tau(\text{Info}(\text{fno}, \text{dep}, \text{arr}, \text{airl})))$

2. $\sigma(\text{Flight}(\text{city}, \text{dest}, \text{airl}, \text{dep}) \land \tau(\text{Geo}(\text{city}, \text{coun}, \text{pop}))) \rightarrow \exists \text{phone} (\tau(\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone})))$

3. $\sigma(\text{Flight}(\text{src}, \text{city}, \text{airl}, \text{dep}) \land \tau(\text{Geo}(\text{city}, \text{coun}, \text{pop}))) \rightarrow \exists \text{phone} (\tau(\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone})))$

New target schema $\tau'$

<table>
<thead>
<tr>
<th>$\tau'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>InfoAirline( airline, city, coun, phone, year )</td>
</tr>
<tr>
<td>InfoJourney( fno, source, dep, dest, arr, airl )</td>
</tr>
</tbody>
</table>

Mapping rules $M'_{\tau\tau'}$

1. $\tau(\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone})) \rightarrow \exists \text{year} \tau'(\text{InfoAirline}(\text{airl}, \text{city}, \text{coun}, \text{phone}, \text{year}))$

2. $\tau(\text{Route}(\text{fno}, \text{src}, \text{dest})) \land \tau(\text{Info}(\text{fno}, \text{dep}, \text{arr}, \text{airl})) \rightarrow \tau'(\text{InfoJourney}(\text{fno}, \text{dep}, \text{dest}, \text{arr}, \text{airl}))$
Example (DE in Flight Domain)

Source schema $\sigma$

- Geo(city, coun, pop)
- Flight(src, dest, airl, dep)

Target schema $\tau$

- Route(fno, src, dest)
- Info(fno, dep, arr, airl)
- Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $\text{Flight}(\text{src}, \text{dest}, \text{airl}, \text{dep}) \rightarrow \exists \text{fno} \exists \text{arr} (\text{Route}(\text{fno}, \text{src}, \text{dest}) \land \text{Info}(\text{fno}, \text{dep}, \text{arr}, \text{airl}))$
2. $\text{Flight}(\text{city}, \text{dest}, \text{airl}, \text{dep}) \land \text{Geo}(\text{city}, \text{coun}, \text{pop}) \rightarrow \exists \text{phone} (\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone}))$
3. $\text{Flight}(\text{src}, \text{city}, \text{airl}, \text{dep}) \land \text{Geo}(\text{city}, \text{coun}, \text{pop}) \rightarrow \exists \text{phone} (\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone}))$
Example (DE in Flight Domain)

Source schema $\sigma$

- $\text{Geo}(\text{city}, \text{coun}, \text{pop})$
- $\text{Flight}(\text{src}, \text{dest}, \text{airl}, \text{dep})$

Target schema $\tau$

- $\text{Route}(\text{fno}, \text{src}, \text{dest})$
- $\text{Info}(\text{fno}, \text{dep}, \text{arr}, \text{airl})$
- $\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone})$

Mapping rules $M_{\sigma \tau}$

1. $\text{Flight}(\text{src}, \text{dest}, \text{airl}, \text{dep}) \rightarrow \exists \text{fno} \exists \text{arr} (\text{Route}(\text{fno}, \text{src}, \text{dest}) \land \text{Info}(\text{fno}, \text{dep}, \text{arr}, \text{airl}))$
2. $\text{Flight}(\text{city}, \text{dest}, \text{airl}, \text{dep}) \land \text{Geo}(\text{city}, \text{coun}, \text{pop}) \rightarrow \exists \text{phone} (\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone}))$
3. $\text{Flight}(\text{src}, \text{city}, \text{airl}, \text{dep}) \land \text{Geo}(\text{city}, \text{coun}, \text{pop}) \rightarrow \exists \text{phone} (\text{Serves}(\text{airl}, \text{city}, \text{coun}, \text{phone}))$

New target schema $\tau'$

- $\text{InfoAirline}(\text{airline}, \text{city}, \text{coun}, \text{phone}, \text{year})$
- $\text{InfoJourney}(\text{fno}, \text{source}, \text{dep}, \text{dest}, \text{arr}, \text{airl})$
Example (DE in Flight Domain)

Source schema $\sigma$

- Geo( city, coun, pop )
- Flight( src, dest, airl, dep )

Target schema $\tau$

- Route( fno, src, dest )
- Info( fno, dep, arr, airl )
- Serves( airl, city, coun, phone )

Mapping rules $M_{\sigma \tau}$

1. $\text{Flight}(src, dest, airl, dep) \rightarrow \exists fno \exists arr (\text{Route}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))$
2. $\text{Flight}(city, dest, airl, dep) \land \text{Geo}(city, coun, pop) \rightarrow \exists phone (\text{Serves}(airl, city, coun, phone))$
3. $\text{Flight}(src, city, airl, dep) \land \text{Geo}(city, coun, pop) \rightarrow \exists phone (\text{Serves}(airl, city, coun, phone))$

New target schema $\tau'$

- InfoAirline( airline, city, coun, phone, year )
- InfoJourney( fno, source, dep, dest, arr, airl )

Mapping rules $M_{\tau \tau'}$

1. $\text{Serves}(airl, city, coun, phone) \rightarrow \exists year \text{InfoAirline}(airl, city, coun, phone, year)$
2. $\text{Route}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl) \rightarrow \text{InfoJourney}(fno, dep, dest, arr, airl)$
Example (DE in Flight Domain)

**Source schema** $\sigma$

- Geo( city, coun, pop )
- Flight( src, dest, airl, dep )

**Target schema** $\tau$

- Route( fno, src, dest )
- Info( fno, dep, arr, airl )
- Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $\text{Flight}(src, dest, airl, dep) \rightarrow \exists fno \exists arr (\text{Route}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl))$
2. $\text{Flight}(\text{city}, dest, airl, dep) \land \text{Geo}(\text{city}, coun, pop) \rightarrow \exists \text{phone} (\text{Serves}(airl, city, coun, phone))$
3. $\text{Flight}(src, \text{city}, airl, dep) \land \text{Geo}(\text{city}, coun, pop) \rightarrow \exists \text{phone} (\text{Serves}(airl, city, coun, phone))$

**New target schema** $\tau'$

- InfoAirline( airline, city, coun, phone, year)
- InfoJourney( fno, source, dep, dest, arr, airl )

**Mapping rules** $M_{\tau\tau'}$

1. $\text{Serves}(airl, city, coun, phone) \rightarrow \exists \text{year} \text{InfoAirline}(airl, city, coun, phone, year)$
2. $\text{Route}(fno, src, dest) \land \text{Info}(fno, dep, arr, airl) \rightarrow \text{InfoJourney}(fno, dep, dest, arr, airl)$

**Composed rules** $M_{\sigma\tau} \circ M_{\tau\tau'}$

1. $\text{Flight}(src, dest, airl, dep) \rightarrow \exists fno \exists arr (\text{InfoJourney}(fno, src, dep, dest, arr, airl))$
2. $\text{Flight}(\text{city}, dest, airl, dep) \land \text{Geo}(\text{city}, coun, pop) \rightarrow \exists \text{phone} \exists \text{year} \text{InfoAirline}(airl, city, coun, phone, year)$
3. $\text{Flight}(src, \text{city}, airl, dep) \land \text{Geo}(\text{city}, coun, pop) \rightarrow \exists \text{phone} \exists \text{year} \text{InfoAirline}(airl, city, coun, phone, year)$
Inverse for Source Schema Change

\[ \sigma \text{DB} \rightarrow \text{Exchange} \rightarrow \tau \text{DB} \]

\[ M_{\sigma \tau} \]

\[ \tau \text{query} \]

\[ \text{target schema } \tau \]

\[ \text{source schema } \sigma \]

\[ \text{materialized} \]
Inverse for Source Schema Change

source schema $\sigma'$

$M_{\sigma\sigma'}$

mapping rules

source schema $\sigma$

$M_{\sigma\tau}$

mapping rules

target schema $\tau$

$\tau$ query

$\sigma'$ DB

$\sigma$ DB

Exchange

materialized

$\tau$ DB
Inverse for Source Schema Change

source schema $\sigma'$

mapping rules $M_{\sigma\sigma'}$

$\sigma'$ DB

exchange

target schema $\tau$

mapping rules $M_{\sigma\tau}$

$\tau$ DB

materialized

$\tau$ query
Inverse for Source Schema Change

Composed mapping \((M_{\sigma\sigma'})^{-1} \circ M_{\tau\tau'}\)

source schema \(\sigma'\)

mapping rules \(M_{\sigma\sigma'}\)

\(\sigma'\) DB

mapping rules \(M_{\sigma\tau}\)

target schema \(\tau\)

\(\tau\) query

Exchange

\(\tau\) DB

materialized
Main question: Closure

- Are mappings closed under
  - composition?
  - inverse?
- In general they are not
- Solution: Use second order logic with Skolem functions
Mapping Composition
- Treat mappings as binary relations

\[ [M_{\tau_1 \tau_2}] = \text{set of pairs (source } \tau_1\text{-instance, } \tau_2\text{-solution)} \]
Definition (Mapping composition)

Given schemata $\sigma, \tau, \tau'$ and mappings $M_{\sigma\tau}, M_{\tau\tau'}$. The composition of $M_{\sigma\tau}, M_{\tau\tau'}$ is defined by

$$[M_{\sigma\tau}] \circ [M_{\tau\tau'}] = \{(\mathcal{S}, \mathcal{I}') \mid \text{there is } \tau\text{-instance } \mathcal{I} \text{ s.t.}$$

$$(\mathcal{S}, \mathcal{I}) \in [M_{\sigma\tau}] \text{ and } (\mathcal{I}, \mathcal{I}') \in [M_{\tau\tau'}]\}$$
Treat mappings as binary relations
\[[\mathcal{M}_{\tau_1\tau_2}] = \text{set of pairs (source } \tau_1\text{-instance, } \tau_2\text{-solution)}\]

**Definition (Mapping composition)**

Given schemata \(\sigma, \tau, \tau'\) and mappings \(\mathcal{M}_{\sigma\tau}, \mathcal{M}_{\tau\tau'}\). The composition of \(\mathcal{M}_{\sigma\tau}, \mathcal{M}_{\tau\tau'}\) is defined by

\[[\mathcal{M}_{\sigma\tau}] \circ [\mathcal{M}_{\tau\tau'}] = \{(\mathcal{G}, \mathcal{I}') \mid \text{there is } \tau\text{-instance } \mathcal{I} \text{ s.t. }\
\quad (\mathcal{G}, \mathcal{I}) \in [\mathcal{M}_{\sigma\tau}] \text{ and } (\mathcal{I}, \mathcal{I}') \in [\mathcal{M}_{\tau\tau'}]\}\]

- Note: Semantics of composition does not say whether there exist rule set \(\mathcal{M}\) representing \([\mathcal{M}_{\sigma\tau}] \circ [\mathcal{M}_{\tau\tau'}]\).
  (That is the whole point of the closure problem)
Example

- \( \sigma : \{ \text{Takes}(\text{name}, \text{course}) \} \)
- \( \tau : \{ \text{Takes1}(\text{name}, \text{course}), \text{Student}(\text{name}, \text{sid}) \} \)
- \( \tau' : \{ \text{Enrolled}(\text{sid}, \text{course}) \} \)
- \( \mathcal{M}_{\sigma \tau} : \{ \text{Takes}(\text{n}, \text{c}) \rightarrow \text{Takes1}(\text{n}, \text{c}), \text{Takes}(\text{n}, \text{c}) \rightarrow \exists \text{s}\text{Student}(\text{n}, \text{s}) \} \)
- \( \mathcal{M}_{\tau \tau'} : \{ \text{Student}(\text{n}, \text{s}) \land \text{Takes1}(\text{n}, \text{c}) \rightarrow \text{Enrolled}(\text{s}, \text{c}) \} \)
Example

- $\sigma: \{\text{Takes(name, course)}\}$
- $\tau: \{\text{Takes1(name, course), Student(name, sid)}\}$
- $\tau': \{\text{Enrolled(sid, course)}\}$
- $M_{\sigma\tau}: \{\text{Takes(n, c) }\rightarrow\text{Takes1(n, c), Takes(n, c) }\rightarrow\exists s \text{Student}(n, s)\}$
- $M_{\tau\tau'}: \{\text{Student(n, s) }\land\text{Takes1(n, c) }\rightarrow\text{Enrolled}(s, c)\}$

- No st-tgd represents $M_{\sigma\tau} \circ M_{\tau\tau'}$, in particular not st-tgd:

  $$\text{Takes}(n, c) \rightarrow \exists y \text{Enrolled}(y, c)$$
Example

- \( \sigma : \{ \text{Takes}(\text{name}, \text{course}) \} \)
- \( \tau : \{ \text{Takes1}(\text{name}, \text{course}), \text{Student}(\text{name}, \text{sid}) \} \)
- \( \tau' : \{ \text{Enrolled}(\text{sid}, \text{course}) \} \)
- \( M_{\sigma \tau} : \{ \text{Takes}(n, c) \rightarrow \text{Takes1}(n, c), \text{Takes}(n, c) \rightarrow \exists s \text{Student}(n, s) \} \)
- \( M_{\tau \tau'} : \{ \text{Student}(n, s) \land \text{Takes1}(n, c) \rightarrow \text{Enrolled}(s, c) \} \)

No st-tgd represents \( M_{\sigma \tau} \circ M_{\tau \tau'} \), in particular not st-tgd:

\[
\text{Takes}(n, c) \rightarrow \exists y \text{Enrolled}(y, c)
\]

Intuitively need to express dependency \( f : n \rightarrow \text{sid} \)

\[
\text{Takes}(n, c) \rightarrow \text{Enrolled}(f(n), c)
\]
Example

- $\sigma : \{\text{Takes}(\text{name}, \text{course})\}$
- $\tau : \{\text{Takes1}(\text{name}, \text{course}), \text{Student}(\text{name}, \text{sid})\}$
- $\tau' : \{\text{Enrolled}(\text{sid}, \text{course})\}$
- $M_{\sigma \tau} :$
  \{\text{Takes}(n, c) \rightarrow \text{Takes1}(n, c), \text{Takes}(n, c) \rightarrow \exists s \text{Student}(n, s)\}$
- $M_{\tau \tau'} : \{\text{Student}(n, s) \land \text{Takes1}(n, c) \rightarrow \text{Enrolled}(s, c)\}$

- No st-tgd represents $M_{\sigma \tau} \circ M_{\tau \tau'}$, in particular not st-tgd:
  \[\text{Takes}(n, c) \rightarrow \exists y \text{Enrolled}(y, c)\]

- Intuitively need to express dependency $f : n \rightarrow \text{sid}$
  \[\text{Takes}(n, c) \rightarrow \text{Enrolled}(f(n), c)\]

- $f$ called Skolem function
Complexity of Relational Composition

Problem \textit{COMPOSITION}(M_{\sigma \tau}, M_{\tau \tau'})

- INPUT: Instance $\mathcal{G}$ of $\sigma$ and instance $\mathcal{F}'$ of $\tau'$
- Output: Is $(\mathcal{G}, \mathcal{F}') \in [M_{\sigma \tau}] \circ [M_{\tau \tau'}]$?
Complexity of Relational Composition

Problem \textsc{Composition}(M_{\sigma \tau}, M_{\tau \tau'})

- INPUT: Instance \( \mathcal{G} \) of \( \sigma \) and instance \( \mathcal{F}' \) of \( \tau' \)
- Output: Is \((\mathcal{G}, \mathcal{F}') \in [M_{\sigma \tau}] \circ [M_{\tau \tau'}]?)

Theorem

- For mappings \( M_{\sigma \tau} \) and \( M_{\tau \tau'} \) specified by st-tgds, \( \textsc{Composition}(M_{\sigma \tau}, M_{\tau \tau'}) \) is NP.
- One can find \( M_{\sigma \tau}^* \) and \( M_{\tau \tau'}^* \), represented by st-tgds for which \( \textsc{Composition}(M_{\sigma \tau}^*, M_{\tau \tau'}^*) \) is NP-complete.

Proof by reducing from NP-hard problem of 3-colorability
Corollary

For the mappings $M^{*}_{\sigma \tau}$ and $M^{*}_{\tau \tau'}$ specified by st-tgds there is no finite set of FOL formulae representing their composition.

Proof sketch

- Assume for contradiction there is set $X$ of FOL formulae for the composition.
- Then the NP-hard $COMPOSITION(M^{*}_{\sigma \tau}, M^{*}_{\tau \tau'})$ reduces to checking $(\mathcal{G}, \mathcal{G}') \models X$
- which is in $AC^0$
- But $AC^0 \subsetneq NP$, \$.
Definition (SO tgds)

Given disjoint schemata $\sigma, \tau$, a second-order tuple-generating dependency from $\sigma$ to $\tau$ is a formula of the form

$$\exists f_1 \ldots \exists f_m (\forall \vec{x}_1 (\phi_1 \rightarrow \psi_1) \land \cdots \land \forall \vec{x}_n (\phi_n \rightarrow \psi_n))$$

where

- each $f_i$ is a function symbol
- each $\phi_i$ is conjunction of relational formulae $R(y_1, \ldots, y_k)$ or identities $t = t'$ with $y_j$ from $\vec{x}$ and $t, t'$ are terms built from $\{\vec{x}_i, f_1, \ldots, f_m\}$
- $\psi_i$ is conjunction of form $R(t_1, \ldots, t_l)$ and $t_j$ built from $\{\vec{x}_i, f_1, \ldots, f_m\}$
- each variable in $\vec{x}_i$ appears in some relational atom of $\phi_i$

$f_1, \ldots, f_m$ are called Skolem functions.
Semantics of SO tgd

- As in second order logic but requiring that \((k\text{-ary})\) \(f\) s are interpreted by \(k\text{-ary}\) functions of form

\[
f : (\text{CONST} \cup \text{VAR})^k \rightarrow \text{CONST} \cup \text{VAR}
\]
SO tgds do the job

Theorem

- For mappings $\mathcal{M}_{\sigma\tau}$ and $\mathcal{M}_{\tau\tau'}$ specified by SO tgds $\Sigma_{\sigma\tau}$, $\Sigma_{\tau\tau'}$, resp., there is a set of SO tgds representing $[\mathcal{M}_{\sigma\tau}] \circ [\mathcal{M}_{\tau\tau'}]$. 
- Moreover there is an exponential-time algorithm computing the composition.
SO tgds do the job

Theorem

- For mappings $M_{\sigma\tau}$ and $M_{\tau\tau'}$ specified by SO tgds $\Sigma_{\sigma\tau}$, $\Sigma_{\tau\tau'}$, resp., there is a set of SO tgds representing $[M_{\sigma\tau}] \circ [M_{\tau\tau'}]$.

- Moreover there is an exponential-time algorithm computing the composition.

- This theorem applicable to mappings described by FOL st-tgds: Transform st-tgds into SO tgds using skolemization.
Composing relational schema mappings

Require: on the source side reuse of variables only in equalities

Input : \( \Sigma_{\sigma \tau}, \Sigma_{\tau \tau}' \)

Output : \( \Sigma_{\sigma \tau}' \)

\( \Sigma_{\sigma \tau}' := \emptyset; \)

\( m := \max_{\phi \rightarrow \psi \in \Sigma_{\tau \tau}'} ||\phi||; \)

forall \( \phi_1 \rightarrow \pi_1, \ldots, \phi_k \rightarrow \pi_k \in \Sigma_{\sigma \tau}, k \leq m \) do

in case of repetitions rename variables;

\( \rho := \pi_1 \land \cdots \land \pi_k; \)

forall \( \pi \land \alpha \rightarrow \pi' \in \Sigma_{\tau \tau}' \) and all homomorphisms \( h : \pi \rightarrow \rho \) do

\( \Sigma_{\sigma \tau}' = \Sigma_{\sigma \tau}' \cup \{ \phi_1 \land \cdots \land \phi_k \land h(\alpha) \rightarrow h(\pi') \} \)

end

end

return \( \Sigma_{\sigma \tau}'; \)

Notation used in algorithm

▷ \( ||\phi|| = \) number of atoms in \( \phi \)

▷ use \( \pi \) for conjunctions of relational atoms and \( \alpha \) for equality atoms

▷ So each SO tgd can be written as \( \pi \land \alpha \rightarrow \pi' \)
Inverting Mappings
First Definition of Inverse

- Harder than composition.
- Intuition: $\mathcal{M} \circ \mathcal{M}^{-1} = \text{“identity mapping” } ID$
- But even semantics not clear: what should $ID$ be?
- Let us start with

**Definition (Inverse)**

The mapping $\mathcal{M}_{\tau\sigma}^{-1}$ is an inverse of mapping $\mathcal{M}_{\sigma\tau}$ iff

$$\mathcal{M}_{\sigma\tau} \circ \mathcal{M}_{\tau\sigma}^{-1} = \{(\mathcal{G}, \mathcal{G}') \mid \mathcal{G}, \mathcal{G}' \text{ are } \sigma\text{-instances with } \mathcal{G} \subseteq \mathcal{G}'\}$$
Example

- Inverses may not be unique
  - $\mathcal{M}_{\sigma \tau} : S(x) \rightarrow T(x), S(x) \rightarrow T'(x)$
  - First inverse $\mathcal{M}_{\tau \sigma}^{-1} : T(x) \rightarrow S(x)$.
  - Another inverse: $\mathcal{M}_{\tau \sigma}^{-1} : T'(x) \rightarrow S(x)$.

- Inverse of union requires disjunction
  - $\mathcal{M}_{\sigma \tau} : S(x) \rightarrow T(x), S'(x) \rightarrow T(x)$
  - $\mathcal{M}_{\tau \sigma}^{-1} : T(x) \rightarrow S(x) \lor S'(x)$
  - So inverse (in some mapping language such as st-tgd) may not exist
    $\implies$ Criteria for existence of inverse mappings
Subset property

**Definition (Subset property)**

Mapping $M_{\sigma\tau}$ satisfies the subset property iff for all pairs $(\mathcal{S}, \mathcal{S}')$:

If $\text{Sol}_{M_{\sigma\tau}}(\mathcal{S}) \subseteq \text{Sol}_{M_{\sigma\tau}}(\mathcal{S}')$ then $\mathcal{S}' \subseteq \mathcal{S}$

**Theorem**

Let $M_{\sigma\tau}$ be specified by a set of st-tgds. Then it is invertible iff it fulfils the subset property.
Complexity of Checking Invertibility

**Theorem**

Let $\mathcal{M}_{\sigma\tau}$ be specified by a set of st-tgds. Checking invertibility is coNP-complete.

Surprisingly the seemingly simpler problem is not decidable:

**Theorem**

Let $\mathcal{M}_{\sigma\tau}$ and $\mathcal{M}'_{\tau\sigma}$ be specified by finite sets of st-tgds. It is undecidable whether $\mathcal{M}'_{\tau\sigma}$ is an inverse of $\mathcal{M}_{\sigma\tau}$
Relaxed Notions of Invertibility

- **Quasi-inverse**
  - Not considered here, because
  - even for this relaxed notion existence of st-tgd mappings not guaranteed

- **We consider notion of (maximum) recover**
  - Recover sound information w.r.t. mappings
  - Existence of covers guaranteed
Definition (Recovery)

A mapping $\mathcal{M}' = \mathcal{M}'_{\tau\sigma}$ is a

- **recovery of mapping** $\mathcal{M} = \mathcal{M}_{\sigma\tau}$ iff for every $\sigma$ instance $\mathcal{S}$ on which $\mathcal{M}$ is defined (for short: $\mathcal{S} \in \text{Dom}(\mathcal{M})$) it holds that $(\mathcal{S}, \mathcal{S}) \in \mathcal{M} \circ \mathcal{M}'$.

- **maximum recovery of mapping** $\mathcal{M}_{\sigma\tau}$ iff it is a recovery and is maximal: for every recovery $\mathcal{M}''$ of $\mathcal{M}$ it holds that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$. 


Definition (Recovery)

A mapping $\mathcal{M}' = \mathcal{M}'_{\tau\sigma}$ is a

- recovery of mapping $\mathcal{M} = \mathcal{M}_{\sigma\tau}$ iff for every $\sigma$ instance $S$ on which $\mathcal{M}$ is defined (for short: $S \in \text{Dom}(\mathcal{M})$) it holds that $(S, S) \in \mathcal{M} \circ \mathcal{M}'$.

- maximum recovery of mapping $\mathcal{M}_{\sigma\tau}$ iff it is a recovery and is maximal: for every recovery $\mathcal{M}''$ of $\mathcal{M}$ it holds that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$

- The smaller the space of possible solutions by inverse $\mathcal{M}'$ the more informative is $\mathcal{M}'$
Example (Recoveries)

- $\sigma$: $\{E(x, y)\}$
- $\tau$: $\{F(x, y), G(x)\}$
- $\mathcal{M} = (\sigma, \tau, \Sigma)$ with

\[
\Sigma = \{E(x, z) \land E(z, y) \rightarrow F(x, y) \land G(z)\}
\]

- $\mathcal{M}_1 = (\tau, \sigma, \Sigma_1)$ with

\[
\Sigma_1 = \{F(x, y) \rightarrow \exists z (E(x, z) \land E(z, y))\}
\]
Example (Recoveries)

- $\sigma$: \{ $E(x, y)$ \}
- $\tau$: \{ $F(x, y), G(x)$ \}
- $\mathcal{M} = (\sigma, \tau, \Sigma)$ with

  \[ \Sigma = \{ E(x, z) \land E(z, y) \rightarrow F(x, y) \land G(z) \} \]

- $\mathcal{M}_1 = (\tau, \sigma, \Sigma_1)$ with

  \[ \Sigma_1 = \{ F(x, y) \rightarrow \exists z(E(x, z) \land E(z, y)) \} \]

- $\mathcal{M}_1$ is a recovery of $\mathcal{M}$
  - For any instance $\mathcal{S}$ let $\mathcal{U}$ be universal canonical solution for $\mathcal{M}$.
  - Then $(\mathcal{U}, \mathcal{S}) \in \mathcal{M}_1$ (so $(\mathcal{S}, \mathcal{S}) \in \mathcal{M} \circ \mathcal{M}_1$)
Example (Recoveries)

- \( \sigma : \{ E(x, y) \} \)
- \( \tau : \{ F(x, y), G(x) \} \)
- \( \mathcal{M} = (\sigma, \tau, \Sigma) \) with

\[
\Sigma = \{ E(x, z) \land E(z, y) \rightarrow F(x, y) \land G(z) \}
\]

- \( \mathcal{M}_2 = (\tau, \sigma, \Sigma_2) \) with

\[
\Sigma_2 = \{ G(z) \rightarrow \exists x, y(E(x, z) \land E(z, y)) \} \]
Example (Recoveries)

- \( \sigma: \{ E(x, y) \} \)
- \( \tau: \{ F(x, y), G(x) \} \)
- \( \mathcal{M} = (\sigma, \tau, \Sigma) \text{ with } \)
  \[
  \Sigma = \{ E(x, z) \land E(z, y) \to F(x, y) \land G(z) \} 
  \]

- \( \mathcal{M}_2 = (\tau, \sigma, \Sigma_2) \text{ with } \)
  \[
  \Sigma_2 = \{ G(z) \to \exists x, y( E(x, z) \land E(z, y) ) \} 
  \]

- \( \mathcal{M}_2 \) is a recovery of \( \mathcal{M} \)
Example (Recoveries)

- $\sigma$: $\{E(x, y)\}$
- $\tau$: $\{F(x, y), G(x)\}$
- $M = (\sigma, \tau, \Sigma)$ with
  \[\Sigma = \{E(x, z) \land E(z, y) \to F(x, y) \land G(z)\}\]

- $M_3 = (\tau, \sigma, \Sigma_3)$ with
  \[\Sigma_3 = \{F(x, y) \land G(z) \to E(x, z) \land E(z, y)\}\]
Example (Recoveries)

- $\sigma$: $\{E(x, y)\}$
- $\tau$: $\{F(x, y), G(x)\}$
- $M = (\sigma, \tau, \Sigma)$ with

\[
\Sigma = \{E(x, z) \land E(z, y) \rightarrow F(x, y) \land G(z)\}
\]

- $M_3 = (\tau, \sigma, \Sigma_3)$ with

\[
\Sigma_3 = \{F(x, y) \land G(z) \rightarrow E(x, z) \land E(z, y)\}
\]

- $M_3$ is not a recovery of $M$
  - See exercise
Example (Recoveries)

- $\sigma$: $\{E(x, y)\}$
- $\tau$: $\{F(x, y), G(x)\}$
- $\mathcal{M} = (\sigma, \tau, \Sigma)$ with

$$\Sigma = \{E(x, z) \land E(z, y) \rightarrow F(x, y) \land G(z)\}$$

- $\mathcal{M}_4 = (\tau, \sigma, \Sigma_4)$ with

$$\Sigma_4 = \Sigma_1 \cup \Sigma_2$$
Example (Recoveries)

- \( \sigma: \{E(x, y)\} \)
- \( \tau: \{F(x, y), G(x)\} \)
- \( \mathcal{M} = (\sigma, \tau, \Sigma) \) with

\[
\Sigma = \{E(x, z) \land E(z, y) \rightarrow F(x, y) \land G(z)\}
\]

- \( \mathcal{M}_4 = (\tau, \sigma, \Sigma_4) \) with

\[
\Sigma_4 = \Sigma_1 \cup \Sigma_2
\]

- \( \mathcal{M}_4 \) is a maximum recovery of \( \mathcal{M} \)
  - can be shown by the following criteria (exercise).
Closure of st-tgds for Maximum Recovery

**Proposition**

Let $M'_{\tau\sigma}$ be a recovery of $M_{\sigma\tau}$. Then $M'_{\tau\sigma}$ is a maximal recovery iff

1. For every $(S, S') \in M \circ M'$: $S' \in \text{Dom}(M)$ and
2. $M = M \circ M' \circ M$.

Using this one can show

**Theorem**

Every mapping specified by a finite set of st-tgds admits a maximum recovery.
Computing Inverses

- Remember algorithms for view rewriting

**Proposition**

Let $\mathcal{M} = (\sigma, \tau, \Sigma)$ with st-tgds $\Sigma$ and $Q$ be a CQ over $\tau$.

- There exists an algorithm QueryRewriting that computes UCQ with equalities $Q_{\text{rew}}$ that is a rewriting of $Q$ over the source (i.e. $\text{cert}_{\mathcal{M}}(Q, \mathcal{S}) = Q_{\text{rew}}(\mathcal{S})$ for all source DBs $\mathcal{S}$).
- The algorithm runs in exponential time and its output is of exponential size in the size of $\Sigma, Q$.

- Based on QueryRewriting can define algorithm MaximumRecovery

**Theorem**

Algorithm MaximumRecovery produces a maximum recovery in exponential time.
Algorithm MaximumRecovery

Input : $M_{\sigma\tau} = (\sigma, \tau, \Sigma)$ with $\Sigma$ finite set of st-tgds
Output: A maximum recovery $M_{\tau\sigma} = (\tau, \sigma, \Gamma)$
$\Gamma := \emptyset$;
forall $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y}) \in \Sigma$ do
  $Q(\vec{x}) := \exists \vec{y} \psi(\vec{x}, \vec{y})$;
  $\alpha(\vec{x}) := \text{QueryRewriting}(M_{\sigma\tau}, Q)$;
  $\Gamma = \Gamma \cup \{ \psi(\vec{x}, \vec{y}) \land C(\vec{x}) \rightarrow \alpha(\vec{x}) \}$ ; // $C$ is predicate testing for constant
end
return $M_{\tau\sigma} = (\tau, \sigma, \Gamma)$;