Data Exchange and Metadata Management

Marcelo Arenas

PUC Chile & U. of Oxford

Dagstuhl, November 2012

Outline of the talk

- Data exchange: An overview of the relational case
- Metadata management
- Composition operator
- Concluding remarks

Outline of the talk

- Data exchange: An overview of the relational case
- Metadata management
- Composition operator
- Concluding remarks

The problem of data exchange

Given: A source schema **S**, a target schema **T** and a specification $\Sigma_{\mathbf{ST}}$ of the relationship between these schemas

Data exchange: Problem of materializing an instance of **T** given an instance of **S**

- Target instance should reflect the source data as accurately as possible, given the constraints imposed by $\Sigma_{\mathbf{ST}}$ and **T**
- It should be efficiently computable
- It should allow one to evaluate queries on the target in a way that is *semantically consistent* with the source data

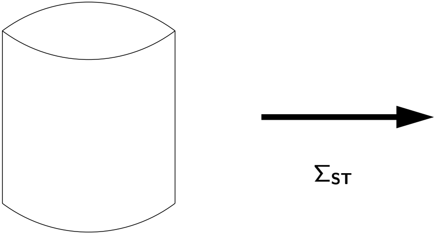
Data exchange in a picture

$\Sigma_{\mathbf{ST}}$

Schema **S**

Schema **T**

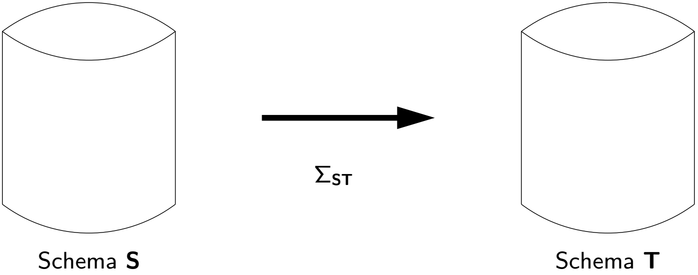
Data exchange in a picture



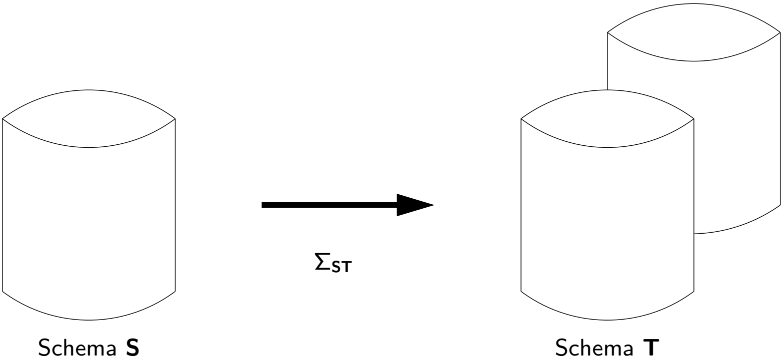
Schema **S**

Schema **T**

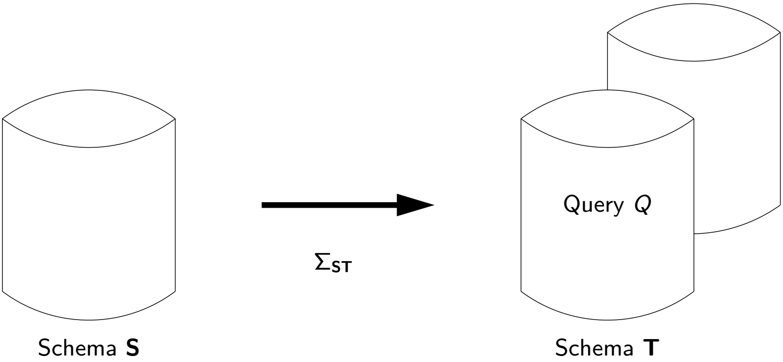
Data exchange in a picture



Data exchange in a picture



Data exchange in a picture



Data exchange: Some fundamental questions

Why is data exchange an interesting problem?

- Is it a difficult problem?

What are the challenges in the area?

- What is a good language for specifying the relationship between source and target data?
- What is a good instance to materialize? Why is it good?
- What does it mean to answer a queries over target data?
- How do we answer queries over target data? Can we do this efficiently?

Data exchange in relational databases

It has been extensively studied in the relational world.

- It has also been implemented: IBM Clio

Relational data exchange setting:

- Source and target schemas: Relational schemas
- Relationship between source and target schemas: Source-to-target tuple-generating dependencies (st-tgds)

Semantics of data exchange has been precisely defined.

- Efficient algorithms for materializing target instances and for answering queries over the target schema have been developed

Schema mapping: The key component in relational data exchange

Schema mapping: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

- **S** and **T** are disjoint relational schemas
- $\Sigma_{\mathbf{ST}}$ is a finite set of st-tgds:

$$\forall \bar{x} \forall \bar{y} \, (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z}))$$

$\varphi(\bar{x}, \bar{y})$: conjunction of relational atomic formulas over **S**

$\psi(\bar{x}, \bar{z})$: conjunction of relational atomic formulas over **T**

Relational data exchange problem

Fixed: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

Problem: Given instance I of $\mathbf{S}$, find an instance J of $\mathbf{T}$ such that (I, J) satisfies $\Sigma_{\mathbf{ST}}$

- (I, J) satisfies $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ if whenever I satisfies $\varphi(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$

Relational data exchange problem

Fixed: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

Problem: Given instance I of $\mathbf{S}$, find an instance J of $\mathbf{T}$ such that (I, J) satisfies $\Sigma_{\mathbf{ST}}$

- (I, J) satisfies $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ if whenever I satisfies $\varphi(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$

Notation

J is a solution for I under $\mathcal{M}$

- $\text{Sol}_{\mathcal{M}}(I)$: Set of solutions for I under $\mathcal{M}$

Relational data exchange: An example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

Relational data exchange: An example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter, 1)$

Relational data exchange: An example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: *dept*(*Peter*,1)

J_2: *dept*(*Peter*,1), *dept*(*Peter*,2)

Relational data exchange: An example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: *dept*(*Peter*,1)

J_2: *dept*(*Peter*,1), *dept*(*Peter*,2)

J_3: *dept*(*Peter*,1), *dept*(*John*,1)

Relational data exchange: An example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\ dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter, 1)$

J_2: $dept(Peter, 1)$, $dept(Peter, 2)$

J_3: $dept(Peter, 1)$, $dept(John, 1)$

J_4: $dept(Peter, n_1)$

Relational data exchange: An example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter, 1)$

J_2: $dept(Peter, 1)$, $dept(Peter, 2)$

J_3: $dept(Peter, 1)$, $dept(John, 1)$

J_4: $dept(Peter, n_1)$

J_5: $dept(Peter, n_1)$, $dept(Peter, n_2)$

Canonical universal solution

Question

What is a good instance to materialize?

Canonical universal solution

Question

What is a good instance to materialize?

Algorithm (chase)

Input : $(\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$ and an instance I of $\mathbf{S}$
Output : Canonical universal solution $J^\star$ for I under $\mathcal{M}$

let $J^\star :=$ empty instance of $\mathbf{T}$
for every $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ in $\Sigma_{\mathbf{ST}}$ **do**
    for every $\bar{a}$, $\bar{b}$ such that I satisfies $\varphi(\bar{a}, \bar{b})$ **do**
        create a fresh tuple $\bar{n}$ of pairwise distinct null values
        insert $\psi(\bar{a}, \bar{n})$ into $J^\star$

Canonical universal solution: Example

Example

Consider mapping $\mathcal{M}$ specified by dependency:

$$employee(x) \quad \rightarrow \quad \exists y\, dept(x, y)$$

Canonical universal solution for $I = \{employee(Peter), employee(John)\}$:

$$J^\star = \{dept(Peter, n_1),\ dept(John, n_2)\}$$

Query answering in data exchange

Given: Mapping $\mathcal{M}$, source instance I and query Q over the target schema

- What does it mean to answer Q?

Query answering in data exchange

Given: Mapping $\mathcal{M}$, source instance I and query Q over the target schema

- What does it mean to answer Q?

Definition (Certain answers)

$$\text{certain}_{\mathcal{M}}(Q, I) \quad = \quad \bigcap_{J \text{ is a solution for } I \text{ under } \mathcal{M}} Q(J)$$

Certain answers: Example

Example

Consider mapping $\mathcal{M}$ specified by:

$$employee(x) \quad \rightarrow \quad \exists y\, dept(x, y)$$

Given instance $I = \{employee(Peter)\}$:

$$\begin{aligned} \text{certain}_{\mathcal{M}}(\exists y\, dept(x, y), I) &= \{Peter\} \\ \text{certain}_{\mathcal{M}}(dept(x, y), I) &= \emptyset \end{aligned}$$

Query rewriting: An approach for answering queries

How can we compute certain answers?

- Naïve algorithm does not work: infinitely many solutions

Query rewriting: An approach for answering queries

How can we compute certain answers?

- Naïve algorithm does not work: infinitely many solutions

Approach proposed in [FKMP03]: **Query Rewriting**

Given a mapping $\mathcal{M}$ and a target query Q, compute a query $Q^\star$ such that for every source instance I with canonical universal solution $J^\star$:

$$\text{certain}_{\mathcal{M}}(Q, I) \quad = \quad Q^\star(J^\star)$$

Query rewriting over the canonical universal solution

Theorem (FKMP03)

Given a mapping $\mathcal{M}$ specified by st-tgds and a union of conjunctive queries Q, there exists a query $Q^\star$ such that for every source instance I with canonical universal solution $J^\star$:

$$\text{certain}_{\mathcal{M}}(Q, I) \;\; = \;\; Q^\star(J^\star)$$

Query rewriting over the canonical universal solution

Theorem (FKMP03)

Given a mapping $\mathcal{M}$ specified by st-tgds and a union of conjunctive queries Q, there exists a query $Q^\star$ such that for every source instance I with canonical universal solution $J^\star$:

$$\text{certain}_{\mathcal{M}}(Q, I) \quad = \quad Q^\star(J^\star)$$

Proof idea: Assume that $\mathbf{C}(a)$ holds whenever a is a constant.

Then:

$$Q^\star(x_1, \ldots, x_m) \quad = \quad \mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_m) \wedge Q(x_1, \ldots, x_m)$$

Computing certain answers: Complexity

Data complexity: Data exchange setting and query are considered to be fixed.

Corollary (FKMP03)

For mappings given by st-tgds, certain answers for **UCQ** *can be computed in polynomial time (data complexity)*

Relational data exchange: Some lessons learned

Key steps in the development of the area:

- Definition of schema mappings: Precise syntax and semantics
  - Definition of the notion of solution
- Identification of good solutions
- Polynomial time algorithms for materializing good solutions
- Definition of target queries: Precise semantics
- Polynomial time algorithms for computing certain answers for **UCQ**

Relational data exchange: Some lessons learned

Key steps in the development of the area:

- Definition of schema mappings: Precise syntax and semantics
  - Definition of the notion of solution
- Identification of good solutions
- Polynomial time algorithms for materializing good solutions
- Definition of target queries: Precise semantics
- Polynomial time algorithms for computing certain answers for **UCQ**

Creating schema mappings is a time consuming and expensive process

- Manual or semi-automatic process in general

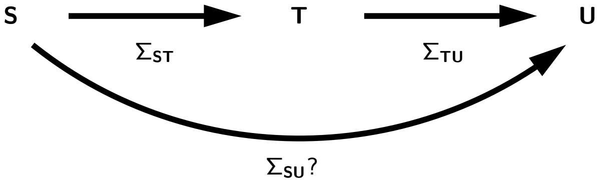
Outline of the talk

- Data exchange: An overview of the relational case
- Metadata management
- Composition operator
- Concluding remarks

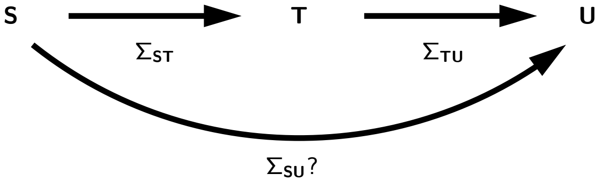
Can we reuse schema mappings?

Can we reuse schema mappings?

Can we reuse schema mappings?

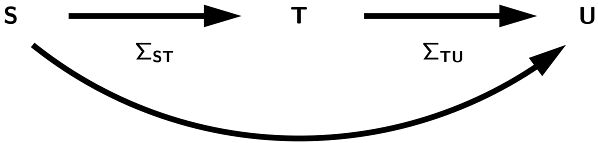


Can we reuse schema mappings?



We need some operators for schema mappings

Can we reuse schema mappings?



$\Sigma_{SU} = \Sigma_{ST} \circ \Sigma_{TU}$

We need some operators for schema mappings

- **Composition** in the above case

Metadata management

Contributions mentioned in the previous slides are just a first step towards the development of a general framework for data exchange.

In fact, as pointed in [B03],

> many information system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation.

Metadata management

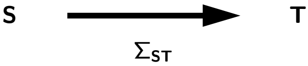
This has motivated the need for the development of a general infrastructure for managing schema mappings.

The problem of managing schema mappings is called **metadata management.**

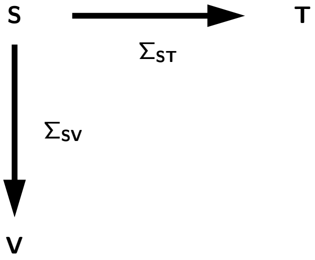
High-level algebraic operators, such as compose, are used to manipulate mappings.

- What other operators are needed?

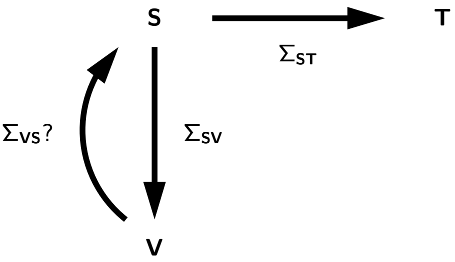
More operators are needed



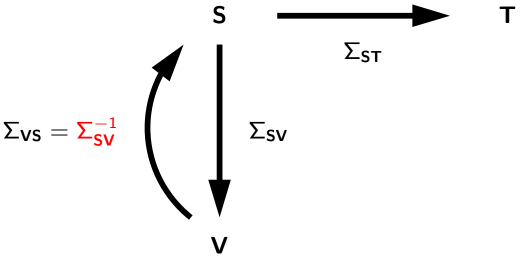
More operators are needed



More operators are needed

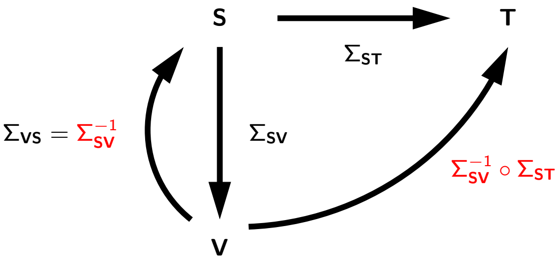


More operators are needed



An **inverse** operator is needed in this case

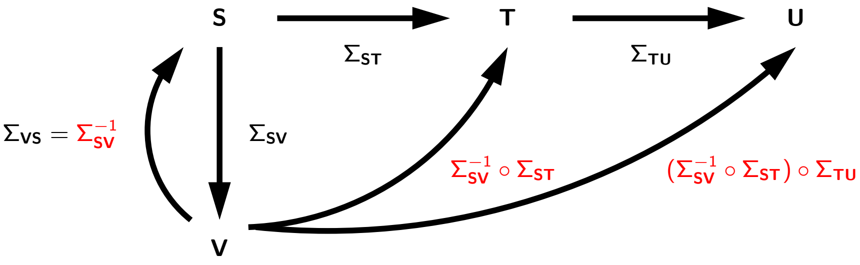
More operators are needed



An **inverse** operator is needed in this case

- Combined with the composition operator

More operators are needed



An **inverse** operator is needed in this case

- Combined with the composition operator

Outline of the talk

- Data exchange: An overview of the relational case
- Metadata management
- Composition operator
- Concluding remarks

The composition operator

Question

What is the semantics of the composition operator?

The composition operator

Question

What is the semantics of the composition operator?

Notation

We can view a mapping $\mathcal{M}$ as a set of pairs:

$$(I, J) \in \mathcal{M} \quad \text{iff} \quad J \in \mathrm{Sol}_{\mathcal{M}}(I)$$

The composition operator

Question

What is the semantics of the composition operator?

Notation

We can view a mapping $\mathcal{M}$ as a set of pairs:

$$(I, J) \in \mathcal{M} \quad \text{iff} \quad J \in \mathrm{Sol}_{\mathcal{M}}(I)$$

Definition (FKPT04)

Let $\mathcal{M}_{12}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{M}_{23}$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_3$:

$$\mathcal{M}_{12} \circ \mathcal{M}_{23} = \{(I_1, I_3) \mid \exists I_2 : (I_1, I_2) \in \mathcal{M}_{12} \text{ and } (I_2, I_3) \in \mathcal{M}_{23}\}$$

Expressing the composition of mappings

Question

What is the right language for expressing the composition?

- st-tgds?

Expressing the composition of mappings

Question

What is the right language for expressing the composition?

- st-tgds?

Example (FKPT04)

Consider mappings:

$$\begin{aligned}
\mathcal{M}_{12} \quad &: \quad takes(n, c) \rightarrow takes_1(n, c) \\
&\quad\;\;\, takes(n, c) \rightarrow \exists s\, student(n, s) \\
\mathcal{M}_{23} \quad &: \quad student(n, s) \wedge takes_1(n, c) \rightarrow enrolled(s, c)
\end{aligned}$$

Expressing the composition of mappings

Question

What is the right language for expressing the composition?

- st-tgds?

Example (FKPT04)

Consider mappings:

$$\begin{aligned} \mathcal{M}_{12} \quad &: \quad takes(n, c) \rightarrow takes_1(n, c) \\ & \quad\;\; takes(n, c) \rightarrow \exists s\, student(n, s) \\ \mathcal{M}_{23} \quad &: \quad student(n, s) \wedge takes_1(n, c) \rightarrow enrolled(s, c) \end{aligned}$$

Does the following st-tgd express the composition?

$$takes(n, c) \quad \rightarrow \quad \exists y\, enrolled(y, c)$$

Example (Cont'd)

This is the right dependency:

$$\forall n \exists y \forall c\, (takes(n, c) \rightarrow enrolled(y, c))$$

Expressing the composition of mappings

Example (Cont'd)

This is the right dependency:

$$\forall n \exists y \forall c\, (takes(n, c) \rightarrow enrolled(y, c))$$

Is first-order logic enough?

- Complexity theory can help us to answer this question

Expressing the composition of mappings: A complexity argument

How difficult is the composition problem?

- Fix mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$
- Problem: Decide whether $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$

If $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is defined by a set of first-order sentences, then the composition problem can be solved efficiently: It is in AC^0

- $\mathrm{AC}^0 \subsetneq \mathrm{PTIME}$

Expressing the composition of mappings: A complexity argument

How difficult is the composition problem?

- Fix mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$
- Problem: Decide whether $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$

If $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is defined by a set of first-order sentences, then the composition problem can be solved efficiently: It is in AC^0

- $AC^0 \subsetneq$ PTIME

But the composition problem is not easy: It can be NP-hard

- $AC^0 \subsetneq \text{PTIME} \subseteq \text{NP}$

Expressing the composition of mappings: A complexity argument

Let see a difficult case taken from [FKPT04].

$\mathcal{M}_{12}$ is specified by:

$$\begin{aligned} node(x) &\rightarrow \exists y\, coloring(x, y) \\ edge(x, y) &\rightarrow edge'(x, y) \end{aligned}$$

$\mathcal{M}_{23}$ is specified by:

$$\begin{aligned} edge'(x, y) \wedge coloring(x, u) \wedge coloring(y, u) &\rightarrow error(x, y) \\ coloring(x, y) &\rightarrow color(y) \end{aligned}$$

Expressing the composition of mappings: A complexity argument

What is the complexity of verifying whether $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$?

Expressing the composition of mappings: A complexity argument

What is the complexity of verifying whether $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$?

Given a graph $G = (N, E)$, consider instances I_1, I_3:

$$\begin{array}{lcl} \textit{node} \text{ in } I_1 & : & N \\ \textit{edge} \text{ in } I_1 & : & E \\ \textit{color} \text{ in } I_3 & : & \{1, 2, 3\} \\ \textit{error} \text{ in } I_3 & : & \emptyset \end{array}$$

Then: G is 3-colorable iff $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$

Back to our initial question:

What is the right language for expressing the composition?

Expressing the composition of mappings

Back to our initial question:

What is the right language for expressing the composition?

Complexity theory can help us again:

- NP-hardness and Fagin's theorem: We need at least existential second-order logic

Expressing the composition of mappings

Back to our initial question:

What is the right language for expressing the composition?

Complexity theory can help us again:

- NP-hardness and Fagin's theorem: We need at least existential second-order logic
- Good news: There is a nice second-order language for expressing the composition

SO tgds: The right language for expressing the composition of mappings

Example

Consider again the mappings:

$$\begin{aligned}\mathcal{M}_{12} \quad &: \quad \mathit{takes}(n, c) \rightarrow \mathit{takes}_1(n, c)\\ &\quad\;\; \mathit{takes}(n, c) \rightarrow \exists s\, \mathit{student}(n, s)\\ \mathcal{M}_{23} \quad &: \quad \mathit{student}(n, s) \wedge \mathit{takes}_1(n, c) \rightarrow \mathit{enrolled}(s, c)\end{aligned}$$

The following SO tgd defines the composition:

$$\exists f \forall n \forall c\, (\mathit{takes}(n, c) \rightarrow \mathit{enrolled}(f(n), c))$$

SO tgds: The right language for expressing the composition of mappings

Example

Consider again mappings $\mathcal{M}_{12}$:

$$\begin{aligned} node(x) &\rightarrow \exists y\, coloring(x, y) \\ edge(x, y) &\rightarrow edge'(x, y) \end{aligned}$$

and $\mathcal{M}_{23}$:

$$\begin{aligned} edge'(x, y) \wedge coloring(x, u) \wedge coloring(y, u) &\rightarrow error(x, y) \\ coloring(x, y) &\rightarrow color(y) \end{aligned}$$

SO tgds: The right language for expressing the composition of mappings

Example (Cont'd)

The following SO tgd defines the composition:

$$\exists f \Big[\forall x(node(x) \to color(f(x))) \land \\ \forall x \forall y(edge(x,y) \land f(x) = f(y) \to error(x,y))\Big]$$

SO tgds: The right language for expressing the composition of mappings

Example (Cont'd)

The following SO tgd defines the composition:

$$\exists f \Big[\forall x (node(x) \to color(f(x))) \wedge \forall x \forall y (edge(x, y) \wedge f(x) = f(y) \to error(x, y)) \Big]$$

This example shows the main ingredients of SO tgds:

- Predicates including terms: $color(f(x))$
- Equality between terms in the premises: $f(x) = f(y)$

SO tgds: The right language for expressing the composition of mappings

SO tgds were introduced in [FKPT04]

- They have good properties regarding data exchange and composition
  - Canonical universal solution and certain answers to **UCQ** can be computed in polynomial time (data complexity)

SO tgds: The right language for expressing the composition of mappings

SO tgds were introduced in [FKPT04]

- They have good properties regarding data exchange and composition
  - Canonical universal solution and certain answers to **UCQ** can be computed in polynomial time (data complexity)

Theorem (FKPT04)

If $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ are specified by SO tgds, then $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can be specified by an SO tgd

SO tgds: The right language for expressing the composition of mappings

SO tgds were introduced in [FKPT04]

- They have good properties regarding data exchange and composition
  - Canonical universal solution and certain answers to **UCQ** can be computed in polynomial time (data complexity)

Theorem (FKPT04)

If $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ are specified by SO tgds, then $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can be specified by an SO tgd

- *There exists an exponential time algorithm that computes such SO tgds*

SO tgds: The right language for expressing the composition of mappings

Corollary (FKPT04)

The composition of a finite number of mappings, each defined by a finite set of st-tgds, is defined by an SO tgd

SO tgds: The right language for expressing the composition of mappings

Corollary (FKPT04)

The composition of a finite number of mappings, each defined by a finite set of st-tgds, is defined by an SO tgd

But not only that, SO tgds are *exactly* the right language:

Theorem (FKPT05)

Every SO tgd defines the composition of a finite number of mappings, each defined by a finite set of st-tgds.

Outline of the talk

- Data exchange: An overview of the relational case
- Metadata management
- Composition operator
- Concluding remarks

Concluding remarks

- Composition and inverse operators are fundamental in metadata management
- The problem of composing schema mappings given by st-tgds is solved
- Considerable progress has been made on the problem of inverting schema mappings
- Combining these operators is an open issue
  - Some progress has been made
  - But we do not know whether there is a good language for both operators. Is there a reasonable language that is closed under both operators?

Thank you!

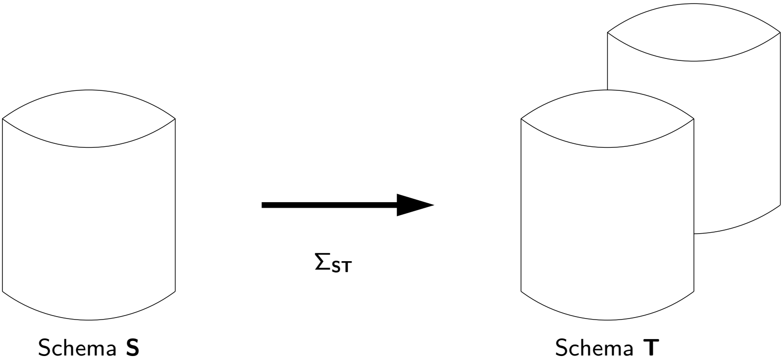
Backup slides

- Inverse operator
- Combination of both operators
  - Key ingredient: Conditional tables

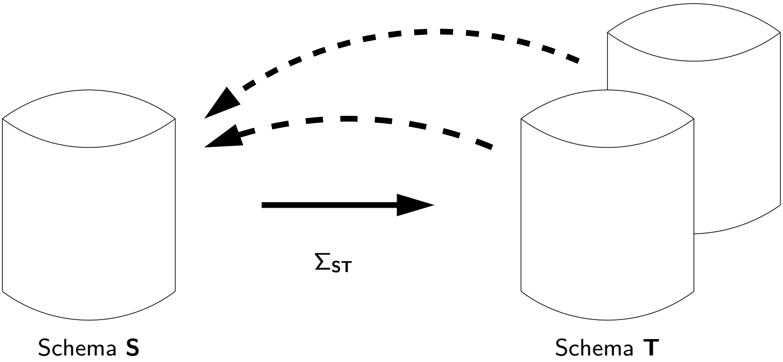
Backup slides

- Inverse operator
- Combination of both operators
  - Key ingredient: Conditional tables

The inverse operator



The inverse operator



The inverse operator

Question

What is the semantics of the inverse operator?

This turns out to be a very difficult question.

We consider three notions of inverse here:

- Fagin-inverse
- Quasi-inverse
- Maximum recovery

The notion of Fagin-inverse

Intuition: A mapping composed with its inverse should be equal to the identity mapping

The notion of Fagin-inverse

Intuition: A mapping composed with its inverse should be equal to the identity mapping

What is the identity mapping?

- $\text{Id}_{\mathbf{S}} \ = \ \{(I, I) \mid I \text{ is an instance of } \mathbf{S}\}$?

The notion of Fagin-inverse

Intuition: A mapping composed with its inverse should be equal to the identity mapping

What is the identity mapping?

- $\text{Id}_{\mathbf{S}} = \{(I, I) \mid I \text{ is an instance of } \mathbf{S}\}$?

For mapping specified by st-tgds, $\text{Id}_{\mathbf{S}}$ is not the right notion.

- $\overline{\text{Id}}_{\mathbf{S}} = \{(I_1, I_2) \mid I_1, I_2 \text{ are instances of } \mathbf{S} \text{ and } I_1 \subseteq I_2\}$

The notion of Fagin-inverse: Formal definition

Definition (F06)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a Fagin-inverse of $\mathcal{M}$ if:

$$\mathcal{M} \circ \mathcal{M}^\star \;=\; \overline{\mathsf{Id}}_{\mathbf{S}_1}$$

The notion of Fagin-inverse: Formal definition

Definition (F06)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a Fagin-inverse of $\mathcal{M}$ if:

$$\mathcal{M} \circ \mathcal{M}^\star \quad = \quad \overline{\mathsf{Id}}_{\mathbf{S}_1}$$

Example

Consider mapping $\mathcal{M}$ specified by:

$$A(x) \quad \rightarrow \quad R(x) \wedge \exists y\, S(x, y)$$

Then the following are Fagin-inverses of $\mathcal{M}$:

$$\begin{array}{lcl} \mathcal{M}_1^\star & : & R(x) \rightarrow A(x) \\ \mathcal{M}_2^\star & : & S(x, y) \rightarrow A(x) \end{array}$$

Is Fagin-inverse the right notion of inverse for mappings?

On the positive side: It is a natural notion

- With good computational properties

On the negative side: A mapping specified by st-tgds is not guaranteed to admit a Fagin-inverse

- For example: Mapping specified by $A(x, y) \rightarrow R(x)$ does not admit a Fagin-inverse

In fact: This notion turns out to be rather restrictive, as it is rare that a schema mapping possesses a Fagin-inverse.

Is Fagin-inverse the right notion of inverse for mappings?

The notion of quasi-inverse was introduced in [FKPT07] to overcome this limitation.

- The idea is to relax the notion of Fagin-inverse by not differentiating between source instances that are equivalent for data exchange purposes

Numerous non-Fagin-invertible mappings possess natural and useful quasi-inverses.

- But there are still simple mappings specified by st-tgds that have no quasi-inverse

The notion of maximum recovery overcome this limitation.

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v\, emp(x, u, v)$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v\, emp(x, u, v) \quad \checkmark$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v\, emp(x, u, v) \quad \checkmark$$

$$\mathcal{M}_2^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u\, emp(x, u, z)$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v\, emp(x, u, v) \quad ✓$$
$$\mathcal{M}_2^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u\, emp(x, u, z) \quad ✓$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$\mathcal{M}_1^\star$: $shuttle(x, z) \rightarrow \exists u \exists v \, emp(x, u, v)$ ✓

$\mathcal{M}_2^\star$: $shuttle(x, z) \rightarrow \exists u \, emp(x, u, z)$ ✓

$\mathcal{M}_3^\star$: $shuttle(x, z) \rightarrow \exists u \, emp(x, z, u)$

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\begin{array}{llcll} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) & \checkmark \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) & \checkmark \\ \mathcal{M}_3^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, z, u) & \times \end{array}$$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{lccl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \end{array}$$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{lccc} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{lccl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \\ \mathcal{M}_4^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \wedge u \neq z \end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{lrcl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \\ \mathcal{M}_4^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \wedge u \neq z \end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$
$\mathcal{M}_4^\star$ is better than $\mathcal{M}_2^\star$ and $\mathcal{M}_1^\star$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{lccl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \\ \mathcal{M}_4^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \wedge u \neq z \end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$
$\mathcal{M}_4^\star$ is better than $\mathcal{M}_2^\star$ and $\mathcal{M}_1^\star$

We would like to find a recovery of $\mathcal{M}$ that is better than any other recovery: Maximum recovery

The notion of recovery: Formalization

Definition (APR08)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$ and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a recovery of $\mathcal{M}$ if:

for every instance I of $\mathbf{S}_1$: $(I, I) \in \mathcal{M} \circ \mathcal{M}^\star$

The notion of recovery: Formalization

Definition (APR08)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$ and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a recovery of $\mathcal{M}$ if:

for every instance I of $\mathbf{S}_1$: $(I, I) \in \mathcal{M} \circ \mathcal{M}^\star$

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

This mapping is not a recovery of $\mathcal{M}$:

$$\mathcal{M}_3^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u\, emp(x, z, u)$$

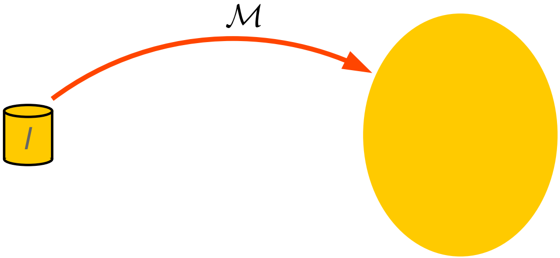
The notion of recovery: Formalization

Example (Cont'd)

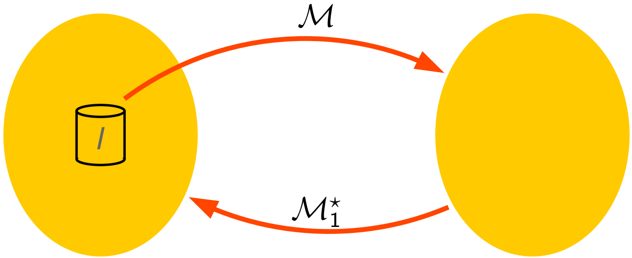
On the other hand, these mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llcl} \mathcal{M}_1^\star: & \mathit{shuttle}(x, z) & \rightarrow & \exists u \exists v\, \mathit{emp}(x, u, v) \\ \mathcal{M}_2^\star: & \mathit{shuttle}(x, z) & \rightarrow & \exists u\, \mathit{emp}(x, u, z) \\ \mathcal{M}_4^\star: & \mathit{shuttle}(x, z) & \rightarrow & \exists u\, \mathit{emp}(x, u, z) \wedge u \neq z \end{array}$$

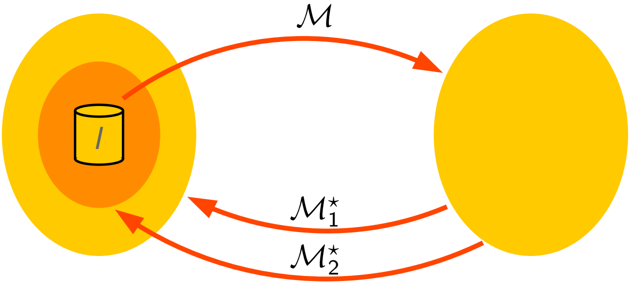
The notion of maximum recovery



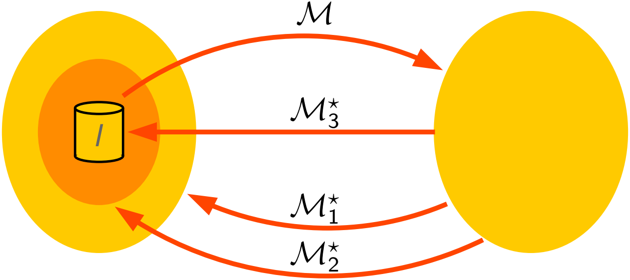
The notion of maximum recovery



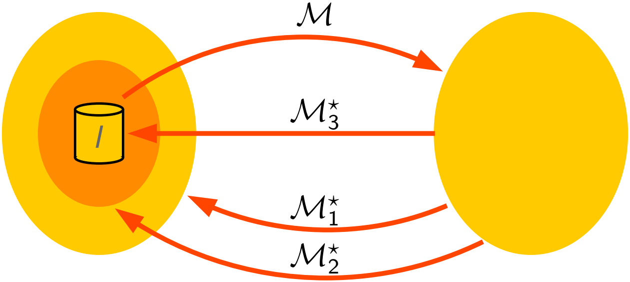
The notion of maximum recovery



The notion of maximum recovery



The notion of maximum recovery



Definition (APR08)

$\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$ if:

- $\mathcal{M}^\star$ is a recovery of $\mathcal{M}$
- for every recovery $\mathcal{M}'$ of $\mathcal{M}$: $\mathcal{M} \circ \mathcal{M}^\star \subseteq \mathcal{M} \circ \mathcal{M}'$

A basic property of (maximum) recoveries

We have seen three notions of inversion for mappings.

- How can we show that a notion of inverse is appropriate?

A basic property of (maximum) recoveries

We have seen three notions of inversion for mappings.

- How can we show that a notion of inverse is appropriate?

A criterion: How much of the initial information is recovered?

A basic property of (maximum) recoveries

We have seen three notions of inversion for mappings.

- How can we show that a notion of inverse is appropriate?

A criterion: How much of the initial information is recovered?

- How close is a space of solution to a particular solution?

A basic property of (maximum) recoveries

We have seen three notions of inversion for mappings.

- How can we show that a notion of inverse is appropriate?

A criterion: How much of the initial information is recovered?

- How close is a space of solution to a particular solution? How close is $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^{\star}}(I)$ to I?

A basic property of (maximum) recoveries

We have seen three notions of inversion for mappings.

- How can we show that a notion of inverse is appropriate?

A criterion: How much of the initial information is recovered?

- How close is a space of solution to a particular solution? How close is $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^{\star}}(I)$ to I?

Simple approach: Compare the information that can be retrieved from I and $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^{\star}}(I)$

A basic property of (maximum) recoveries

To compare the information that can be retrieved from I and $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^\star}(I)$: Compare $Q(I)$ to $\text{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I)$

A basic property of (maximum) recoveries

To compare the information that can be retrieved from I and $\mathrm{Sol}_{\mathcal{M} \circ \mathcal{M}^\star}(I)$: Compare $Q(I)$ to $\mathrm{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I)$

Observation

Let $\mathcal{M}$ be a mapping from **S** to **T**, I an instance of **S**, Q a query over **S** and $\mathcal{M}^\star$ a recovery of $\mathcal{M}$:

$$\mathrm{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I) \quad \subseteq \quad Q(I)$$

A basic property of (maximum) recoveries

To compare the information that can be retrieved from I and $\text{Sol}_{\mathcal{M}\circ\mathcal{M}^\star}(I)$: Compare $Q(I)$ to $\text{certain}_{\mathcal{M}\circ\mathcal{M}^\star}(Q, I)$

Observation

Let $\mathcal{M}$ be a mapping from **S** to **T**, I an instance of **S**, Q a query over **S** and $\mathcal{M}^\star$ a recovery of $\mathcal{M}$:

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

Information retrieved from $\text{Sol}_{\mathcal{M}\circ\mathcal{M}^\star}(I)$ is sound w.r.t. I.

A basic property of (maximum) recoveries

To compare the information that can be retrieved from I and $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^\star}(I)$: Compare $Q(I)$ to $\text{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I)$

Observation

Let $\mathcal{M}$ be a mapping from **S** to **T**, I an instance of **S**, Q a query over **S** and $\mathcal{M}^\star$ a recovery of $\mathcal{M}$:

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

Information retrieved from $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^\star}(I)$ is sound w.r.t. I.

- Is $\text{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I) = Q(I)$?

A basic property of (maximum) recoveries

To compare the information that can be retrieved from I and $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^\star}(I)$: Compare $Q(I)$ to $\text{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I)$

Observation

Let $\mathcal{M}$ be a mapping from **S** to **T**, I an instance of **S**, Q a query over **S** and $\mathcal{M}^\star$ a recovery of $\mathcal{M}$:

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

Information retrieved from $\text{Sol}_{\mathcal{M} \circ \mathcal{M}^\star}(I)$ is sound w.r.t. I.

- Is $\text{certain}_{\mathcal{M} \circ \mathcal{M}^\star}(Q, I) = Q(I)$?
- Not always possible: $P(x, y) \rightarrow R(x)$ and $Q(x, y) = P(x, y)$

A fundamental property of maximum recoveries

Definition

- $\mathcal{M}'$ recovers Q under $\mathcal{M}$ if for every source instance I:

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

- Q can be recovered under $\mathcal{M}$ if the above mapping $\mathcal{M}'$ exists

A fundamental property of maximum recoveries

Definition

- $\mathcal{M}'$ recovers Q under $\mathcal{M}$ if for every source instance I:

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

- Q can be recovered under $\mathcal{M}$ if the above mapping $\mathcal{M}'$ exists

Theorem (APRR09)

Let $\mathcal{M}^\star$ be a maximum recovery of a mapping $\mathcal{M}$. If Q can be recovered under $\mathcal{M}$, then $\mathcal{M}^\star$ recovers Q under $\mathcal{M}$.

On the existence of maximum recoveries

Maximum recoveries overcome one of the limitations of Fagin-inverses and quasi-inverses.

On the existence of maximum recoveries

Maximum recoveries overcome one of the limitations of Fagin-inverses and quasi-inverses.

Theorem (APR08)

Every mapping specified by st-tgds has a maximum recovery.

On the existence of maximum recoveries

Maximum recoveries overcome one of the limitations of Fagin-inverses and quasi-inverses.

Theorem (APR08)

Every mapping specified by st-tgds has a maximum recovery.

Example

Consider a mapping $\mathcal{M}$ specified by:

$$P(x, y) \wedge P(y, z) \quad \rightarrow \quad R(x, z) \wedge T(y)$$

$\mathcal{M}$ has neither an inverse nor a quasi-inverse [FKPT07]. A maximum recovery of $\mathcal{M}$ is specified by:

$$\begin{aligned} R(x, z) \quad &\rightarrow \quad \exists y\, P(x, y) \wedge P(y, z) \\ T(y) \quad &\rightarrow \quad \exists x \exists z\, P(x, y) \wedge P(y, z) \end{aligned}$$

Maximum recoveries strictly generalize Fagin-inverses

$\mathcal{M}$ is closed-down on the left if it satisfies the following condition:

If J is a solution for I_2 and $I_1 \subseteq I_2$, then J is a solution for I_1

The notion of Fagin-inverse is defined in [F06] focusing on these mappings.

Maximum recoveries strictly generalize Fagin-inverses

$\mathcal{M}$ is closed-down on the left if it satisfies the following condition:

If J is a solution for I_2 and $I_1 \subseteq I_2$, then J is a solution for I_1

The notion of Fagin-inverse is defined in [F06] focusing on these mappings.

Theorem (APR08)

If $\mathcal{M}$ is closed-down on the left and Fagin-invertible: $\mathcal{M}^\star$ is an inverse of $\mathcal{M}$ iff $\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$.

Maximum recoveries strictly generalize Fagin-inverses

$\mathcal{M}$ is closed-down on the left if it satisfies the following condition:

If J is a solution for I_2 and $I_1 \subseteq I_2$, then J is a solution for I_1

The notion of Fagin-inverse is defined in [F06] focusing on these mappings.

Theorem (APR08)

If $\mathcal{M}$ is closed-down on the left and Fagin-invertible: $\mathcal{M}^\star$ is an inverse of $\mathcal{M}$ iff $\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$.

A *similar* theorem can be proved for the notion of quasi-inverse.

Computing maximum recoveries

The simple process of "reversing the arrows" of st-tgds does not work properly

- For example, consider mapping specified by st-tgds $A(x) \rightarrow T(x)$ and $B(x) \rightarrow T(x)$

Computing maximum recoveries

The simple process of "reversing the arrows" of st-tgds does not work properly

- For example, consider mapping specified by st-tgds $A(x) \to T(x)$ and $B(x) \to T(x)$

We present an algorithm that is based on query rewriting.

- We can reuse the large body of work on query rewriting

Definition

Given a mapping $\mathcal{M}$ and a target query Q: Query Q' is a rewriting over the source of Q if for every source instance I:

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

Computing maximum recoveries

Algorithm

Input : A mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds

Output : A mapping $\mathcal{M}^\star = (\mathbf{T}, \mathbf{S}, \Sigma^\star)$ that is a maximum recovery of $\mathcal{M}$

let $\Sigma^\star := \emptyset$

for every $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{y})$ in Σ **do**

compute a first-order logic formula $\alpha(\bar{x})$ that is a source rewriting of $\exists \bar{z}\, \psi(\bar{x}, \bar{z})$ under $\mathcal{M}$

add dependency $\psi(\bar{x}, \bar{z}) \wedge \mathbf{C}(\bar{x}) \rightarrow \alpha(\bar{x})$ to $\Sigma^\star$

Complexity of the algorithm

Theorem (APR08,APR09)

There is an exponential time algorithm that, given a mapping $\mathcal{M}$ specified by st-tgds, computes a maximum recovery of $\mathcal{M}$.

Complexity of the algorithm

Theorem (APR08,APR09)

There is an exponential time algorithm that, given a mapping $\mathcal{M}$ specified by st-tgds, computes a maximum recovery of $\mathcal{M}$.

A few words about the language needed to express the maximum recovery:

- Output of the algorithm: $\mathbf{CQ}^{\mathbf{C}(\cdot)}$**-to-**$\mathbf{UCQ}^{=}$ dependencies
- Predicate $\mathbf{C}(\cdot)$, disjunction and equality are needed

Backup slides

- Inverse operator
- Combination of both operators
  - Key ingredient: Conditional tables

We need to combine the operators

Can we combine the composition and inverse operators?

- Is there a good language for both operators?

We need to combine the operators

Can we combine the composition and inverse operators?

- Is there a good language for both operators?

Some bad news:

Theorem (APR11)

There exists a mapping specified by an SO tgd that has neither a Fagin-inverse nor a quasi-inverse nor a maximum recovery.

We need to combine the operators

Can we combine the composition and inverse operators?

- Is there a good language for both operators?

Some bad news:

Theorem (APR11)

There exists a mapping specified by an SO tgd that has neither a Fagin-inverse nor a quasi-inverse nor a maximum recovery.

Do we need yet another notion of inverse?

We need to combine the operators

Can we combine the composition and inverse operators?

- Is there a good language for both operators?

Some bad news:

Theorem (APR11)

There exists a mapping specified by an SO tgd that has neither a Fagin-inverse nor a quasi-inverse nor a maximum recovery.

Do we need yet another notion of inverse?

- No, we need to revisit the semantics of mappings

What went wrong?

Key observation: A target instance of a mapping can be the source instance of another mapping.

- Sources instances may contain null values

What went wrong?

Key observation: A target instance of a mapping can be the source instance of another mapping.

- Sources instances may contain null values

Example

Consider a mapping $\mathcal{M}$ specified by:

$$P(x,y) \rightarrow R(x,y)$$
$$P(x,x) \rightarrow T(x)$$

The canonical universal solution for $I = \{P(n,a)\}$ under $\mathcal{M}$:

$$J^\star = \{R(n,a)\}$$

But $J^\star$ is not a *good* solution for I.

- It cannot represent the fact that if n is given value a, then $T(a)$ should hold in the target.

A solution to the problem

We use conditional tables instead of the usual instances.

- What about complexity?

A solution to the problem

We use conditional tables instead of the usual instances.

- What about complexity?

Example

Consider again mapping $\mathcal{M}$ specified by:

$$\begin{array}{lcl} P(x,y) & \rightarrow & R(x,y) \\ P(x,x) & \rightarrow & T(x) \end{array}$$

The following conditional table is a good solution for $I = \{P(n,a)\}$:

$R(n,a)$	$true$
$T(n)$	$n = a$

Can conditional tables be used in *real* data exchange systems?

Good news: We just need positive conditions

- Good solutions can be computed in polynomial time (data complexity)
- Certain answers for **UCQ** can be computed in polynomial time (data complexity)

Can conditional tables be used in *real* data exchange systems?

Good news: We just need positive conditions

- Good solutions can be computed in polynomial time (data complexity)
- Certain answers for **UCQ** can be computed in polynomial time (data complexity)

Theorem (APR11)

If instances are replaced by positive conditional tables:

- *SO tgds are still the right language for the composition of mappings given by st-tgds*
- *Every mapping specified by an SO tgd admits a maximum recovery*

Bibliography

[FKMP03] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa. Data Exchange: Semantics and Query Answering. ICDT 2003: 207-224

[B03] P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. CIDR 2003

[FKPT04] R. Fagin, P. G. Kolaitis, L. Popa, W.-C. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. PODS 2004: 83-94

[FKPT05] R. Fagin, P. G. Kolaitis, L. Popa, W.-C. Tan. Composing schema mappings: Second-order dependencies to the rescue. TODS 30(4): 994-1055, 2005

[F06] R. Fagin. Inverting schema mappings. PODS 2006: 50-59

Bibliography

[FKPT07] R. Fagin, P. G. Kolaitis, L. Popa, W.-C. Tan. Quasi-inverses of schema mappings. PODS 2007: 123-132

[APR08] M. Arenas, J. Pérez, C. Riveros. The recovery of a schema mapping: bringing exchanged data back. PODS 2008: 13-22

[APRR09] M. Arenas, J. Pérez, J. Reutter, C. Riveros. Inverting Schema Mappings: Bridging the Gap between Theory and Practice. PVLDB 2(1): 1018-1029, 2009

[APR09] M. Arenas, J. Pérez, C. Riveros: The recovery of a schema mapping: Bringing exchanged data back. TODS 34(4), 2009

[APR11] M. Arenas, J. Pérez, J. Reutter. Data Exchange beyond Complete Data. PODS 2011: 83-94