Data Exchange beyond Complete Data

Marcelo Arenas

Department of Computer Science
Pontificia Universidad Católica de Chile

Joint work with Jorge Pérez (U. de Chile) and Juan Reutter (U. Edinburgh)
The need for a more general data exchange framework
  - Two important scenarios: Incomplete databases and knowledge bases

Formalism for exchanging representations systems

Applications to incomplete databases

Applications to metadata management

Concluding remarks
The need for a more general data exchange framework
  - Two important scenarios: Incomplete databases and knowledge bases

Formalism for exchanging representations systems

Applications to incomplete databases

Applications to metadata management

Concluding remarks
Key steps in the development of the area:

- Definition of schema mapping: Precise syntax and semantics
- Definition of the notion of solution
- Identification of good solutions
  - Universal solutions
- Polynomial time algorithms for materializing good solutions
  - Based on the chase procedure
Data exchange: Some lessons learned

Key steps in the development of the area:

- Definition of schema mapping: Precise syntax and semantics
- Definition of the notion of solution
- Identification of good solutions
  - Universal solutions
- Polynomial time algorithms for materializing good solutions
  - Based on the chase procedure

Creating schema mappings is a time consuming and expensive process

- Manual or semi-automatic process in general
Ongoing project: Reusing schema mappings

\[ \Sigma_{ST} \]
Ongoing project: Reusing schema mappings
Ongoing project: Reusing schema mappings
Ongoing project: Reusing schema mappings

We need some operators for schema mappings
Ongoing project: Reusing schema mappings

\[ \Sigma_{SU} = \Sigma_{ST} \circ \Sigma_{TU} \]

We need some operators for schema mappings

- **Composition** in the above case
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?
An inverse operator is also needed
An inverse operator is also needed
An inverse operator is also needed

\[ \Sigma_{VS} \]

\[ \Sigma_{ST} \]

\[ \Sigma_{SV} \]

\[ \Sigma_{V} \]
An inverse operator is also needed

\[ \Sigma_{VS} = \Sigma_{SV}^{-1} \]
An inverse operator is also needed

\[ \Sigma_{\text{VS}} = \Sigma_{\text{SV}}^{-1} \]

\[ \Sigma_{\text{SV}} \]

\[ \Sigma_{\text{ST}} \]

\[ \Sigma_{\text{SV}}^{-1} \circ \Sigma_{\text{ST}} \]

Composition and inverse operators have to be combined.
An inverse operator is also needed

\[ \Sigma_{VS} = \Sigma_{SV}^{-1} \]

Composition and inverse operators have to be combined

\[ \Sigma_{SV}^{-1} \circ \Sigma_{ST} \]

\[ (\Sigma_{SV}^{-1} \circ \Sigma_{ST}) \circ \Sigma_{TU} \]
Metadata management: A more general data exchange framework is needed

Composition and inverse operators have been extensively studied in the relational world.

- Semantics, computation, …

Combining these operators is an open issue.
Metadata management: A more general data exchange framework is needed

Composition and inverse operators have been extensively studied in the relational world.

- Semantics, computation, ... 

Combining these operators is an open issue.

- Key observation: A target instance of a mapping can be the source instance of another mapping
Metadata management: A more general data exchange framework is needed

Composition and inverse operators have been extensively studied in the relational world.

▶ Semantics, computation, . . .

Combining these operators is an open issue.

▶ Key observation: A target instance of a mapping can be the source instance of another mapping
▶ Sources instances may contain null values
Metadata management: A more general data exchange framework is needed

Composition and inverse operators have been extensively studied in the relational world.

- Semantics, computation, …

Combining these operators is an open issue.

- Key observation: A target instance of a mapping can be the source instance of another mapping
- Sources instances may contain null values

There is a need for a data exchange framework that can handle databases with incomplete information.
Data exchange in the RDF world

There is an increasing interest in publishing relational data as RDF

- Resulted in the creation of the W3C RDB2RDF Working Group

The problem of translating relational data into RDF can be seen as a data exchange problem

- Schema mappings can be used to describe how the relational data is to be mapped into RDF
There is an increasing interest in publishing relational data as RDF

- Resulted in the creation of the W3C RDB2RDF Working Group

The problem of translating relational data into RDF can be seen as a data exchange problem

- Schema mappings can be used to describe how the relational data is to be mapped into RDF

But there is a mismatch here: A relational database under a **closed-world semantics** is to be translated into an RDF graph under an **open-world semantics**

- There is a need for a data exchange framework that can handle both databases with complete and incomplete information
An issue discussed at the W3C RDB2RDF Working Group: Is a mapping information preserving?

- In particular: For the default mapping defined by this group

How can we address this issue?

- Metadata management can help us
An issue discussed at the W3C RDB2RDF Working Group: Is a mapping information preserving?

- In particular: For the default mapping defined by this group

How can we address this issue?

- Metadata management can help us

Question to answer: Is a mapping invertible?
Data exchange in the RDF world

An issue discussed at the W3C RDB2RDF Working Group: Is a mapping information preserving?

- In particular: For the default mapping defined by this group

How can we address this issue?

- Metadata management can help us

Question to answer: Is a mapping invertible?

- This time an RDF graph is to be translated into a relational database!
Data exchange in the RDF world

An issue discussed at the W3C RDB2RDF Working Group: Is a mapping information preserving?

▶ In particular: For the default mapping defined by this group

How can we address this issue?

▶ Metadata management can help us

Question to answer: Is a mapping invertible?

▶ This time an RDF graph is to be translated into a relational database!

▶ We want to have a unifying framework for all these cases
But these are not the only reasons . . .

Nowadays several applications use knowledge bases to represent data.

- A knowledge base has not only data but also **rules** that allows to infer new data

- In the Semantics Web: RDFS and OWL ontologies
But these are not the only reasons . . .

Nowadays several applications use knowledge bases to represent data.

- A knowledge base has not only data but also rules that allows to infer new data.
- In the Semantics Web: RDFS and OWL ontologies

In a data exchange application over the Semantics Web:

The input is a mapping and a source specification including data and rules, and the output is a target specification also including data and rules.
But these are not the only reasons ... 

Nowadays several applications use knowledge bases to represent data.

- A knowledge base has not only data but also rules that allows to infer new data
- In the Semantics Web: RDFS and OWL ontologies

In a data exchange application over the Semantics Web:

The input is a mapping and a source specification including data and rules, and the output is a target specification also including data and rules

There is a need for a data exchange framework that can handle knowledge bases.
One can exchange more than complete data

- In data exchange one starts with a database instance (with complete information).

- What if we have an initial object that has several interpretations?
  - A representation of a set of possible instances

- We propose a new general formalism to exchange representations of possible instances
  - We apply it to the problems of exchanging instances with incomplete information and exchanging knowledge bases
Outline

- The need for a more general data exchange framework
  - Two important scenarios: Incomplete databases and knowledge bases

- Formalism for exchanging representations systems

- Applications to incomplete databases

- Applications to metadata management

- Concluding remarks
A representation system $\mathcal{R} = (\mathbf{W}, \text{rep})$ consists of:

- a set $\mathbf{W}$ of representatives
- a function $\text{rep}$ that assigns a set of instances to every element in $\mathbf{W}$

$$\text{rep}(\mathbf{V}) = \{I_1, I_2, I_3, \ldots\} \text{ for every } \mathbf{V} \in \mathbf{W}$$

Uniformity assumption: For every $\mathbf{V} \in \mathbf{W}$, there exists a relational schema $\mathbf{S}$ (the type of $\mathbf{V}$) such that $\text{rep}(\mathbf{V}) \subseteq \text{Inst}(\mathbf{S})$
A representation system $\mathcal{R} = (\mathcal{W}, \text{rep})$ consists of:

- a set $\mathcal{W}$ of representatives
- a function rep that assigns a set of instances to every element in $\mathcal{W}$

$$\text{rep}(\mathcal{V}) = \{I_1, I_2, I_3, \ldots\} \text{ for every } \mathcal{V} \in \mathcal{W}$$

Uniformity assumption: For every $\mathcal{V} \in \mathcal{W}$, there exists a relational schema $S$ (the type of $\mathcal{V}$) such that $\text{rep}(\mathcal{V}) \subseteq \text{Inst}(S)$

Incomplete instances and knowledge bases are representation systems
In classical data exchange we consider only *complete* data

\[ \mathcal{M} \text{ is a mapping from } S \text{ to } T \text{ if } \mathcal{M} \subseteq \text{Inst}(S) \times \text{Inst}(T) \]

- Given instances \( I \) of \( S \) and \( J \) of \( T \): \( J \) is a solution for \( I \) under \( \mathcal{M} \) if \( S \) if \( (I, J) \in \mathcal{M} \)
In classical data exchange we consider only *complete* data

\( \mathcal{M} \) is a mapping from \( S \) to \( T \) if \( \mathcal{M} \subseteq \text{Inst}(S) \times \text{Inst}(T) \)

- Given instances \( I \) of \( S \) and \( J \) of \( T \): \( J \) is a solution for \( I \) under \( \mathcal{M} \) if \( S \) if \( (I, J) \in \mathcal{M} \)

\( \mathcal{M} \) is defined by a set \( \Sigma \) of dependencies (e.g., st-tgds) if: \( (I, J) \in \mathcal{M} \) iff \( (I, J) \models \Sigma \).

- Notation: \( \mathcal{M} = (S, T, \Sigma) \)
Extending the definition to representation systems

\( \text{Sol}_M(I) \): Set of solutions for \( I \) under \( M \)
Extending the definition to representation systems

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

This can be extended to set of instances. Given $\mathcal{X} \subseteq \text{Inst}(S)$:

$$\text{Sol}_M(\mathcal{X}) = \bigcup_{I \in \mathcal{X}} \text{Sol}_M(I)$$
Extending the definition to representation systems

Given:

- a mapping $\mathcal{M}$ from $S$ to $T$
- a representation system $\mathcal{R} = (\mathcal{W}, \text{rep})$
- $\mathcal{U}, \mathcal{V} \in \mathcal{W}$ of types $S$ and $T$, respectively
Extending the definition to representation systems

Given:

- a mapping $\mathcal{M}$ from $S$ to $T$
- a representation system $\mathcal{R} = (\mathcal{W}, \text{rep})$
- $\mathcal{U}, \mathcal{V} \in \mathcal{W}$ of types $S$ and $T$, respectively

Definition (APR11)

$\mathcal{V}$ is an $\mathcal{R}$-solution of $\mathcal{U}$ under $\mathcal{M}$ if

$$\text{rep}(\mathcal{V}) \subseteq \text{Sol}_\mathcal{M}(\text{rep}(\mathcal{U}))$$
Extending the definition to representation systems

Given:

- a mapping $M$ from $S$ to $T$
- a representation system $R = (W, \text{rep})$
- $U, V \in W$ of types $S$ and $T$, respectively

Definition (APR11)

$V$ is an $R$-solution of $U$ under $M$ if

\[
\text{rep}(V) \subseteq \text{Sol}_M(\text{rep}(U))
\]

Or equivalently: $V$ is an $R$-solution of $U$ if for every $J \in \text{rep}(V)$, there exists $I \in \text{rep}(U)$ such that $J \in \text{Sol}_M(I)$. 
Extending the definition to representation systems

$$\text{rep}(U) \ni I_1 \ni J_1 \ni \mathcal{M} \ni J_2 \ni I_3 \ni \text{rep}(V)$$
Extending the definition to representation systems

\[ \mathcal{U} \xrightarrow{\mathcal{M}} \mathcal{V} \]

\[ \text{rep}(\mathcal{U}) \]

\[ \text{rep}(\mathcal{V}) \]

\[ l_1, l_2, l_3, l_4 \]

\[ J_1, J_2, J_3 \]
Extending the definition to representation systems

\[ \mathcal{U} \xrightarrow{\mathcal{M}} \mathcal{V} \]

\[ \text{rep}(\mathcal{U}) \xrightarrow{\mathcal{J}_1} \text{rep}(\mathcal{V}) \]

\[ I_1, I_2, I_3, I_4 \]

\[ J_1, J_2, J_3 \]
Extending the definition to representation systems

\[ \mathcal{U} \xrightarrow{\mathcal{M}} \mathcal{V} \]

\[ \text{rep}(\mathcal{U}) \quad \text{rep}(\mathcal{V}) \]

\[ I_1 \quad I_2 \quad I_3 \quad I_4 \]

\[ J_1 \quad J_2 \quad J_3 \]
Extending the definition to representation systems

\[ \mathcal{U} \xrightarrow{\mathcal{M}} \mathcal{V} \]

\[ \text{rep}(\mathcal{U}) \]

\[ \text{rep}(\mathcal{V}) \]

\[ I_1 \]

\[ I_2 \]

\[ I_3 \]

\[ I_4 \]

\[ J_1 \]

\[ J_2 \]

\[ J_3 \]
Universal solutions

What is a good solution in this framework?
What is a good solution in this framework?

**Definition (APR11)**

\( \mathcal{V} \) is an *universal* \( \mathcal{R} \)-solution of \( \mathcal{U} \) under \( \mathcal{M} \) if

\[
\text{rep}(\mathcal{V}) = \text{Sol}_\mathcal{M}(\text{rep}(\mathcal{U}))
\]
Universal solutions in a figure

\[ \mathcal{U} \xrightarrow{\mathcal{M}} \mathcal{V} \]

\[ \text{rep}(\mathcal{U}) \xrightarrow{\text{rep}(\mathcal{V})} \]

\( I_1, I_2, I_3, I_4 \)

\( J_1, J_2, J_3 \)
Strong representation systems

Let $C$ be a class of mappings.
Let $C$ be a class of mappings.

**Definition (APR11)**

$\mathcal{R} = (\mathcal{W}, \text{rep})$ is a **strong representation system** for $C$ if for every $\mathcal{M} \in C$ and for every $U \in \mathcal{W}$, there exists a $V \in \mathcal{W}$:

$$\text{rep}(V) = \text{Sol}_\mathcal{M}(\text{rep}(U))$$
Let $\mathcal{C}$ be a class of mappings.

**Definition (APR11)**

$\mathcal{R} = (\mathcal{W}, \text{rep})$ is a *strong representation system* for $\mathcal{C}$ if for every $\mathcal{M} \in \mathcal{C}$ from $\mathbf{S}$ to $\mathbf{T}$, and for every $\mathcal{U} \in \mathcal{W}$, there exists a $\mathcal{V} \in \mathcal{W}$:

$$\text{rep}(\mathcal{V}) = \text{Sol}_\mathcal{M}(\text{rep}(\mathcal{U}))$$
Let $C$ be a class of mappings.

**Definition (APR11)**

$R = (W, \text{rep})$ is a *strong representation system* for $C$ if for every $M \in C$ from $S$ to $T$, and for every $U \in W$ of type $S$, there exists a $V \in W$:

\[
\text{rep}(V) = \text{Sol}_M(\text{rep}(U))
\]
Let $C$ be a class of mappings.

**Definition (APR11)**

$\mathcal{R} = (\mathcal{W}, \text{rep})$ is a *strong representation system* for $C$ if for every $\mathcal{M} \in C$ from $S$ to $T$, and for every $\mathcal{U} \in \mathcal{W}$ of type $S$, there exists a $\mathcal{V} \in \mathcal{W}$ of type $T$:

$$\text{rep}(\mathcal{V}) = \text{Sol}_\mathcal{M}(\text{rep}(\mathcal{U}))$$
Let $C$ be a class of mappings.

**Definition (APR11)**

$R = (W, \text{rep})$ is a strong representation system for $C$ if for every $M \in C$ from $S$ to $T$, and for every $U \in W$ of type $S$, there exists a $V \in W$ of type $T$:

$$\text{rep}(V) = \text{Sol}_M(\text{rep}(U))$$

If $R = (W, \text{rep})$ is a strong representation system, then the universal solutions for the representatives in $W$ can be represented in the same system.
The need for a more general data exchange framework
  Two important scenarios: Incomplete databases and knowledge bases

Formalism for exchanging representations systems

Applications to incomplete databases

Applications to metadata management

Concluding remarks
Motivating questions

What is a strong representation system for the class of mappings specified by st-tgds?

▷ Are instances including nulls enough?

Can the fundamental data exchange problems be solved in polynomial time in this setting?

▷ Computing (universal) solutions
Naive instances

We have already considered **naive instances**: Instances with null values

▶ Example: Universal solutions

A naive instance $\mathcal{I}$ has labeled nulls:

\[
R(1, n_1) \\
R(n_1, 2) \\
R(1, n_2)
\]
Naive instances

We have already considered **naive instances**: Instances with null values

- Example: Universal solutions

A naive instance $\mathcal{I}$ has labeled nulls:

\[
R(1, n_1) \\
R(n_1, 2) \\
R(1, n_2)
\]

The interpretations of $\mathcal{I}$ are constructed by replacing nulls by constants:

\[
\text{rep}(\mathcal{I}) = \{ K \mid \mu(\mathcal{I}) \subseteq K \text{ for some valuation } \mu \}\]
Are naive instances expressive enough?

Naive instances have been extensively used in data exchange:

**Proposition (FKMP03)**

Let \( \mathcal{M} = (S, T, \Sigma) \), where \( \Sigma \) is a set of st-tgds. Then for every instance \( I \) of \( S \), there exists a naive instance \( J \) of \( T \) such that:

\[
rep(J) = Sol_{\mathcal{M}}(I)
\]

In fact, every universal solution satisfies the property mentioned above.
Are naive instances expressive enough?

But naive instances are not expressive enough to deal with incomplete information in the source instances:

Proposition (APR11)

Naive instances are not a strong representation system for the class of mappings specified by st-tgds
Are naive instances expressive enough?

Example

Consider a mapping \( \mathcal{M} \) specified by:

\[
\begin{align*}
\text{Manager}(x, y) & \rightarrow \text{Reports}(x, y) \\
\text{Manager}(x, x) & \rightarrow \text{SelfManager}(x)
\end{align*}
\]

The \textit{canonical} universal solution for \( \mathcal{I} = \{\text{Manager}(n, \text{Peter})\} \) under \( \mathcal{M} \):

\[
\mathcal{J} = \{\text{Reports}(n, \text{Peter})\}
\]

But \( \mathcal{J} \) is not a \textit{good} solution for \( \mathcal{I} \).

- It cannot represent the fact that if \( n \) is given value Peter, then \( \text{SelfManager}(\text{Peter}) \) should hold in the target.
Conditional instances

What should be added to naive instances to obtain a strong representation system?
Conditional instances

What should be added to naive instances to obtain a strong representation system?

- Answer from database theory: Conditions on the nulls
What should be added to naive instances to obtain a strong representation system?

- Answer from database theory: Conditions on the nulls

Conditional instances: Naive instances plus *tuple conditions*

A tuple condition is a positive Boolean combinations of:

- equalities and inequalities between nulls, and between nulls and constants
Conditional instances

Example

\[
\begin{align*}
R(1, n_1) & : n_1 = n_2 \\
R(n_1, n_2) & : n_1 \neq n_2 \lor n_2 = 2
\end{align*}
\]
Conditional instances

Example

\[
\begin{align*}
R(1, n_1) & \quad n_1 = n_2 \\
R(n_1, n_2) & \quad n_1 \neq n_2 \lor n_2 = 2
\end{align*}
\]

Semantics:
Conditional instances

Example

\[
R(1, n_1) \quad n_1 = n_2 \\
R(n_1, n_2) \quad n_1 \neq n_2 \lor n_2 = 2
\]

Semantics:

\[
\mu(n_1) = \mu(n_2) = 2 \\
\mu(n_1) = \mu(n_2) = 3 \\
\mu(n_1) = 2, \mu(n_2) = 3
\]
Conditional instances

Example

\[
\begin{align*}
R(1, n_1) & \quad n_1 = n_2 \\
R(n_1, n_2) & \quad n_1 \neq n_2 \lor n_2 = 2
\end{align*}
\]

Semantics:

\[
\begin{align*}
\mu(n_1) = \mu(n_2) &= 2 & R(1, 2) \\
R(1, 2) & \quad \mu(n_1) = \mu(n_2) = 3 \\
R(2, 2) & \quad \mu(n_1) = 2, \mu(n_2) = 3
\end{align*}
\]
Conditional instances

Example

\[
\begin{align*}
R(1, n_1) & \quad | \quad n_1 = n_2 \\
R(n_1, n_2) & \quad | \quad n_1 \neq n_2 \lor n_2 = 2
\end{align*}
\]

Semantics:

\[
\frac{\mu(n_1) = \mu(n_2) = 2}{R(1, 2)} \quad \frac{\mu(n_1) = \mu(n_2) = 3}{R(1, 3)} \quad \mu(n_1) = 2, \mu(n_2) = 3
\]

\[
\frac{\mu(n_1) = \mu(n_2) = 3}{R(2, 2)}
\]
Conditional instances

Example

\[
\begin{align*}
R(1, n_1) & \quad n_1 = n_2 \\
R(n_1, n_2) & \quad n_1 \neq n_2 \lor n_2 = 2
\end{align*}
\]

Semantics:

\[
\begin{align*}
\mu(n_1) = \mu(n_2) &= 2 & & \mu(n_1) = \mu(n_2) = 3 & & \mu(n_1) = 2, \mu(n_2) = 3 \\
R(1, 2) & & R(1, 3) & & R(2, 3)
\end{align*}
\]
Conditional instances

Example

\[ R(1, n_1) \quad n_1 = n_2 \]
\[ R(n_1, n_2) \quad n_1 \neq n_2 \lor n_2 = 2 \]

Semantics:

\[ \mu(n_1) = \mu(n_2) = 2 \]
\[ R(1, 2) \]
\[ R(2, 2) \]

\[ \mu(n_1) = \mu(n_2) = 3 \]
\[ R(1, 3) \]

\[ \mu(n_1) = 2, \mu(n_2) = 3 \]
\[ R(2, 3) \]

Interpretations of a conditional instance \( I \):

\[ \text{rep}(I) = \{ K \mid \mu(I) \subseteq K \text{ for some valuation } \mu \} \]
Positive conditional instances

Many problems are intractable over conditional instances.

- We also consider a restricted class of conditional instances

Positive conditional instances: Conditional instances without inequalities
(Positive) conditional instances are enough

**Theorem (APR11)**

*Both conditional instances and positive conditional instances are strong representation systems for the class of mappings specified by st-tgds.*

**Example**

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

$$
\begin{align*}
\text{Manager}(x, y) &\rightarrow \text{Reports}(x, y) \\
\text{Manager}(x, x) &\rightarrow \text{SelfManager}(x)
\end{align*}
$$

The following is a universal solution for $\mathcal{I} = \{\text{Manager}(n, Peter)\}$

$$
\begin{align*}
\text{Reports}(n, Peter) &\quad \text{true} \\
\text{SelfManager}(\text{Peter}) &\quad n = \text{Peter}
\end{align*}
$$
Positive conditional instances are exactly the needed representation system

Theorem (APR11)

All the following are needed to obtain a strong representation system for the class of mappings specified by st-tgds:
Positive conditional instances are exactly the needed representation system

Theorem (APR11)

All the following are needed to obtain a strong representation system for the class of mappings specified by st-tgds:

- equalities between nulls
Positive conditional instances are exactly the needed representation system

**Theorem (APR11)**

All the following are needed to obtain a strong representation system for the class of mappings specified by st-tgds:

- equalities between nulls
  - There exists a mapping $\mathcal{M}$ given by st-tgds and a source naive instance $\mathcal{I}$ such that for every target positive conditional $\mathcal{J}$ not mentioning equalities between nulls: $\text{rep}(\mathcal{J}) \neq \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$
Positive conditional instances are exactly the needed representation system

Theorem (APR11)

All the following are needed to obtain a strong representation system for the class of mappings specified by st-tgds:

- equalities between nulls
  - There exists a mapping $\mathcal{M}$ given by st-tgds and a source naive instance $\mathcal{I}$ such that for every target positive conditional $\mathcal{J}$ not mentioning equalities between nulls: $\text{rep}(\mathcal{J}) \neq \text{Sol}_\mathcal{M}(\text{rep}(\mathcal{I}))$

- equalities between constant and nulls
Positive conditional instances are exactly the needed representation system

Theorem (APR11)

All the following are needed to obtain a strong representation system for the class of mappings specified by st-tgds:

- equalities between nulls
  - There exists a mapping $\mathcal{M}$ given by st-tgds and a source naive instance $I$ such that for every target positive conditional $J$ not mentioning equalities between nulls: $\text{rep}(J) \neq \text{Sol}_M(\text{rep}(I))$

- equalities between constant and nulls

- conjunctions and disjunctions
Positive conditional instances are exactly the needed representation system.

**Theorem (APR11)**

All the following are needed to obtain a strong representation system for the class of mappings specified by st-tgds:

- equalities between nulls
  - There exists a mapping $\mathcal{M}$ given by st-tgds and a source naive instance $I$ such that for every target positive conditional $J$ not mentioning equalities between nulls: $\text{rep}(J) \neq \text{Sol}_\mathcal{M}(\text{rep}(I))$

- equalities between constant and nulls

- conjunctions and disjunctions

Conditional instances are enough but not minimal.
Positive conditional instance can be used in practice!

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\Sigma$ is a set of st-tgds.
Positive conditional instance can be used in practice!

Let $\mathcal{M} = (S, T, \Sigma)$, where $\Sigma$ is a set of st-tgds.

**Theorem (APR11)**

There exists a polynomial time algorithm that, given a positive conditional instance $\mathcal{I}$ over $S$, computes a positive conditional instance $\mathcal{J}$ over $T$ that is a universal solution for $\mathcal{I}$ under $\mathcal{M}$.
Let $\mathcal{M} = (S, T, \Sigma)$, where $\Sigma$ is a set of st-tgds.

**Theorem (APR11)**

*There exists a polynomial time algorithm that, given a positive conditional instance $\mathcal{I}$ over $S$, computes a positive conditional instance $\mathcal{J}$ over $T$ that is a universal solution for $\mathcal{I}$ under $\mathcal{M}$.***

**Remark**

They are also appropriate for query answering in data exchange.

- Same polynomial-time cases as in the usual setting
The need for a more general data exchange framework
  ▶ Two important scenarios: Incomplete databases and knowledge bases

Formalism for exchanging representations systems

Applications to incomplete databases

Applications to metadata management

Concluding remarks
The composition operator

**Definition (FKPT04)**

Let $\mathcal{M}_{12}$ be a mapping from $S_1$ to $S_2$, and $\mathcal{M}_{23}$ a mapping from $S_2$ to $S_3$:

$$\mathcal{M}_{12} \circ \mathcal{M}_{23} = \{(l_1, l_3) \mid \exists l_2 : (l_1, l_2) \in \mathcal{M}_{12} \text{ and } (l_2, l_3) \in \mathcal{M}_{23}\}$$
Expressing the composition of mappings

Question
What is the right language for expressing the composition?
► st-tgds?

Example (FKPT04)
Consider the mappings $\mathcal{M}_{12}$:

- $\text{node}(x) \rightarrow \exists y \ \text{coloring}(x, y)$
- $\text{edge}(x, y) \rightarrow \text{edge}'(x, y)$

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

- $\text{edge}'(x, y) \land \text{coloring}(x, u) \land \text{coloring}(y, u) \rightarrow \text{error}(x, y)$
- $\text{coloring}(x, y) \rightarrow \text{color}(y)$
The following dependency defines the composition:

\[
\exists f \left[ \forall x (\text{node}(x) \rightarrow \text{color}(f(x))) \land \forall x \forall y (\text{edge}(x, y) \land f(x) = f(y) \rightarrow \text{error}(x, y)) \right]
\]
Example (Cont’d)

The following dependency defines the composition:

\[ \exists f \left[ \forall x (\text{node}(x) \rightarrow color(f(x))) \land \forall x \forall y (\text{edge}(x,y) \land f(x) = f(y) \rightarrow \text{error}(x,y)) \right] \]

This example shows the main ingredients of SO tgds:

- Predicates including terms: \( color(f(x)) \)
- Equality between terms: \( f(x) = f(y) \)
SO tgds: The right language for expressing the composition of mappings

SO tgds were introduced in [FKPT04]

- They have good properties regarding composition
SO tgds: The right language for expressing the composition of mappings

SO tgds were introduced in [FKPT04]

- They have good properties regarding composition

**Theorem (FKPT04)**

If $M_{12}$ and $M_{23}$ are specified by SO tgds, then $M_{12} \circ M_{23}$ 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 composition

**Theorem (FKPT04)**

If $M_{12}$ and $M_{23}$ are specified by SO tgds, then $M_{12} \circ M_{23}$ can be specified by an SO tgd
  - There exists an exponential time algorithm that computes such SO tgds
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.
The inverse operator

Schema $S$ \[\Sigma_{ST}\] Schema $T$
The inverse operator

\[ \Sigma_{ST} \]

Schema \( S \) \hspace{1cm} \Sigma_{ST} \hspace{1cm} \text{Schema } T
The inverse operator

Question
What is the semantics of the inverse operator?

This turns out to be a very difficult question.

Several notions of inverse have been considered:

- Fagin-inverse [F06]
- Quasi-inverse [FKPT07]
- Maximum recovery [APR08]
- Maximum extended recovery [FKPT09]
- $C$-maximum recovery [APRR09]
The inverse operator

Question

What is the semantics of the inverse operator?

This turns out to be a very difficult question.

Several notions of inverse have been considered:

- Fagin-inverse [F06]
- Quasi-inverse [FKPT07]
- Maximum recovery [APR08]
- Maximum extended recovery [FKPT09]
- $C$-maximum recovery [APRR09]
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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ 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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

**Example**

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

$$\text{emp}(x, y, z) \land y \neq z \rightarrow \text{shuttle}(x, z)$$

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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

Example

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

\[
emp(x, y, z) \land y \neq z \rightarrow shuttle(x, z)
\]

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

$\mathcal{M}^*_1$: $shuttle(x, z) \rightarrow \exists u \exists v emp(x, u, v)$
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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

**Example**

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

$$\text{emp}(x, y, z) \land y \neq z \rightarrow \text{shuttle}(x, z)$$

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

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

✓
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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

**Example**

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

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

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

- $\mathcal{M}_1^*$: $shuttle(x, z) \rightarrow \exists u \exists v \ emp(x, u, v)$
  - ✓
- $\mathcal{M}_2^*$: $shuttle(x, z) \rightarrow \exists u \ emp(x, u, z)$
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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

Example

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

$$\text{emp}(x, y, z) \land y \neq z \rightarrow \text{shuttle}(x, z)$$

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

- $\mathcal{M}_1^*$: $\text{shuttle}(x, z) \rightarrow \exists u \exists v \text{ emp}(x, u, v)$
  - ✓
- $\mathcal{M}_2^*$: $\text{shuttle}(x, z) \rightarrow \exists u \text{ emp}(x, u, z)$
  - ✓
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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

**Example**

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

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

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

- $\mathcal{M}_1^*$: $shuttle(x, z) \rightarrow \exists u \exists v \; emp(x, u, v)$ ✓
- $\mathcal{M}_2^*$: $shuttle(x, z) \rightarrow \exists u \; emp(x, u, z)$ ✓
- $\mathcal{M}_3^*$: $shuttle(x, z) \rightarrow \exists u \; emp(x, z, u)$
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 $\mathcal{M}^*$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^*$ is called a recovery of $\mathcal{M}$

**Example**

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

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

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

- $\mathcal{M}_{1^*}^*: shuttle(x, z) \rightarrow \exists u \exists v \ emp(x, u, v)$ ✓
- $\mathcal{M}_{2^*}^*: shuttle(x, z) \rightarrow \exists u \ emp(x, u, z)$ ✓
- $\mathcal{M}_{3^*}^*: shuttle(x, z) \rightarrow \exists u \ emp(x, z, u)$ ×
Maximum recovery: The *most informative* recovery

**Example**

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

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

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

- $\mathcal{M}_1^* : shuttle(x, z) \rightarrow \exists u \exists v \ emp(x, u, v)$
- $\mathcal{M}_2^* : shuttle(x, z) \rightarrow \exists u \ emp(x, u, z)$
Maximum recovery: The most informative recovery

Example

Consider again mapping $M$ specified by:

$$\text{emp}(x, y, z) \land y \neq z \rightarrow \text{shuttle}(x, z)$$

These mappings are recoveries of $M$:

- $M_1^*$: $\text{shuttle}(x, z) \rightarrow \exists u \exists v \text{emp}(x, u, v)$
- $M_2^*$: $\text{shuttle}(x, z) \rightarrow \exists u \text{emp}(x, u, z)$

Intuitively: $M_2^*$ is better than $M_1^*$
Maximum recovery: The *most informative* recovery

**Example**

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

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

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

- $\mathcal{M}_1^*: shuttle(x, z) \rightarrow \exists u \exists v \ emp(x, u, v)$
- $\mathcal{M}_2^*: shuttle(x, z) \rightarrow \exists u \ emp(x, u, z)$
- $\mathcal{M}_4^*: shuttle(x, z) \rightarrow \exists u \ emp(x, u, z) \land u \neq z$

Intuitively: $\mathcal{M}_2^*$ is better than $\mathcal{M}_1^*$
Maximum recovery: The most informative recovery

**Example**

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

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

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

- $\mathcal{M}^*_1$: $\text{shuttle}(x, z) \rightarrow \exists u \exists v \ emp(x, u, v)$
- $\mathcal{M}^*_2$: $\text{shuttle}(x, z) \rightarrow \exists u \ emp(x, u, z)$
- $\mathcal{M}^*_4$: $\text{shuttle}(x, z) \rightarrow \exists u \ emp(x, u, z) \land u \neq z$

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

$\mathcal{M}^*_4$ is better than $\mathcal{M}^*_2$ and $\mathcal{M}^*_1$
Maximum recovery: The *most informative* recovery

**Example**

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

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

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

- $\mathcal{M}_1^*: shuttle(x, z) \rightarrow \exists u \exists v \ emp(x, u, v)$
- $\mathcal{M}_2^*: shuttle(x, z) \rightarrow \exists u \ emp(x, u, z)$
- $\mathcal{M}_4^*: shuttle(x, z) \rightarrow \exists u \ emp(x, u, z) \land u \neq z$

Intuitively:

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

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 $S_1$ to $S_2$ and $\mathcal{M}^*$ a mapping from $S_2$ to $S_1$. Then $\mathcal{M}^*$ is a recovery of $\mathcal{M}$ if:

for every instance $I$ of $S_1$: $(I, I) \in \mathcal{M} \circ \mathcal{M}^*$
The notion of recovery: Formalization

**Definition (APR08)**

Let $\mathcal{M}$ be a mapping from $S_1$ to $S_2$ and $\mathcal{M}^*$ a mapping from $S_2$ to $S_1$. Then $\mathcal{M}^*$ is a recovery of $\mathcal{M}$ if:

$$\text{for every instance } I \text{ of } S_1: (I, I) \in \mathcal{M} \circ \mathcal{M}^*$$

**Example**

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

$$\text{emp}(x, y, z) \land y \neq z \rightarrow \text{shuttle}(x, z)$$

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

$$\mathcal{M}_3^*: \text{shuttle}(x, z) \rightarrow \exists u \text{ emp}(x, z, u)$$
The notion of recovery: Formalization

Example (Cont’d)

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

- $\mathcal{M}_1^* : \text{shuttle}(x, z) \rightarrow \exists u \exists v \text{ emp}(x, u, v)$
- $\mathcal{M}_2^* : \text{shuttle}(x, z) \rightarrow \exists u \text{ emp}(x, u, z)$
- $\mathcal{M}_4^* : \text{shuttle}(x, z) \rightarrow \exists u \text{ emp}(x, u, z) \land u \neq z$
The notion of maximum recovery
The notion of maximum recovery
The notion of maximum recovery

\[ \mathcal{M} \]

\[ \mathcal{M}_1^* \]

\[ \mathcal{M}_2^* \]
The notion of maximum recovery
The notion of maximum recovery

Definition (APR08)

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

- $\mathcal{M}^*$ is a recovery of $\mathcal{M}$
- for every recovery $\mathcal{M}'$ of $\mathcal{M}$: $\mathcal{M} \circ \mathcal{M}^* \subseteq \mathcal{M} \circ \mathcal{M}'$
On the existence of maximum recoveries

Theorem (APR08)

Every mapping specified by a finite set of st-tgds has a maximum recovery.
On the existence of maximum recoveries

**Theorem (APR08)**

*Every mapping specified by a finite set of st-tgds has a maximum recovery.*

But this does not hold if one also considers naive instances in the source.

- Maximum extended recovery was introduced to overcome this limitation
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?

Bad news:

**Theorem (APR11)**

*There exists a mapping specified by an SO tgd that does not have a maximum recovery.*
We need to combine the operators

Even worse:

- Previous mapping has neither a Fagin-inverse nor a quasi-inverse nor a $C$-maximum recovery ($\text{CQ} \subseteq C$)
- Semantics of maximum extended recovery is appropriate for st-tgds.
We need to combine the operators

Even worse:

- Previous mapping has neither a Fagin-inverse nor a quasi-inverse nor a $C$-maximum recovery ($\text{CQ} \subseteq C$)
- Semantics of maximum extended recovery is appropriate for st-tgds.

Do we need yet another notion of inverse?
We need to combine the operators

Even worse:

- Previous mapping has neither a Fagin-inverse nor a quasi-inverse nor a $C$-maximum recovery ($CQ \subseteq C$)
- Semantics of maximum extended recovery is appropriate for st-tgds.

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

**Theorem (APR11)**

*Positive conditional instances are a strong representation system for the class of mappings specified by SO tgds.*
A solution to the problem

Theorem (APR11)

If (usual) instances are replaced by positive conditional instances:

- 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
The need for a more general data exchange framework
  - Two important scenarios: Incomplete databases and knowledge bases

Formalism for exchanging representations systems

Applications to incomplete databases

Applications to metadata management

Concluding remarks
We can exchange more than complete data

We propose a general formalism to exchange representation systems

- Applications to incomplete instances
- Applications to metadata management
- Applications to knowledge bases
We can exchange more than complete data

We propose a general formalism to exchange *representation systems*

- Applications to incomplete instances
- Applications to metadata management
- Applications to knowledge bases

Next step: Apply our general setting to the Semantic Web

- Semantic Web data has *nulls* (blank nodes)
- Semantic Web specifications have rules (RDFS, OWL)
Thank you!
Bibliography


<table>
<thead>
<tr>
<th>Reference</th>
<th>Authors</th>
<th>Title</th>
<th>Conference/Year</th>
<th>Pages</th>
</tr>
</thead>
</table>