Exchanging more than Complete Data

Marcelo Arenas

Department of Computer Science
Pontificia Universidad Católica de Chile

This is joint work with Jorge Pérez (U. de Chile) and Juan Reutter (U. Edinburgh)

Outline: First part

- The data exchange problem
  - Some fundamental results in relational data exchange
- The need for a more general data exchange framework
  - Two important scenarios: Incomplete databases and knowledge bases

Outline: First part

- The data exchange problem
  - Some fundamental results in relational data exchange

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

The problem of data exchange

Given: A source schema **S**, a target schema **T** and a specification Σ 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 Σ 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

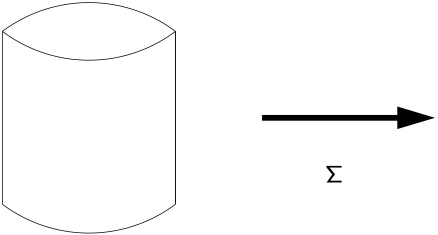
Data exchange in a picture

$\xrightarrow[\Sigma]{}$

Schema **S**

Schema **T**

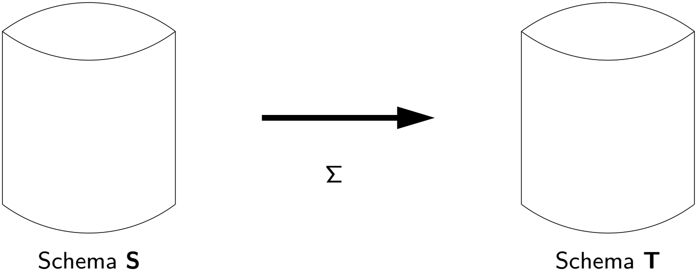
Data exchange in a picture



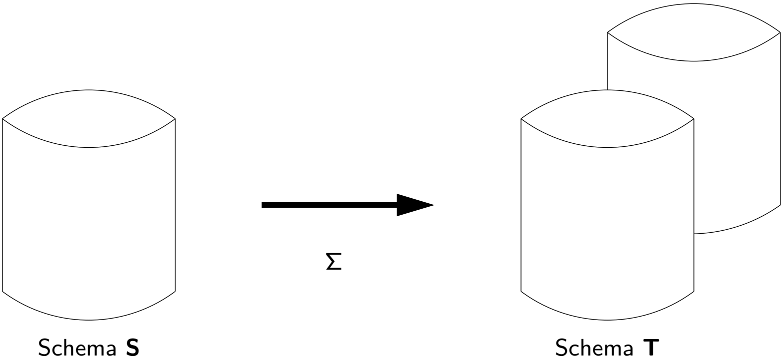
Schema **S**

Schema **T**

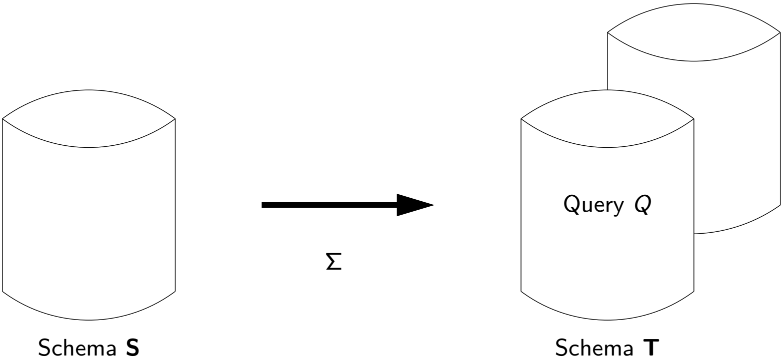
Data exchange in a picture



Data exchange in a picture



Data exchange in a picture



Data exchange: Some fundamental questions

What are the challenges in the area?

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

Exchanging relational data

The data exchange problem has been extensively studied in the relational world.

- It has also been commercially 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: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$

- $\mathbf{S}$ and $\mathbf{T}$ are disjoint relational schemas
- Σ 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 $\mathbf{S}$

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

Relational schema mappings: An example

Example

- **S**: $\texttt{Employee}(\texttt{name})$
- **T**: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ:

$$\forall x \left(\texttt{Employee}(x) \rightarrow \exists y \, \texttt{Dept}(x, y) \right)$$

Relational schema mappings: An example

Example

- **S**: $\texttt{Employee}(\texttt{name})$
- **T**: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ:

$$\forall x \left(\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y) \right)$$

Note

We omit universal quantifiers in st-tgds:

$$\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y)$$

Relational data exchange problem

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

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

- (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: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$

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

- (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 $\mathcal{M}$

The notion of solution: Example

Example

- $\mathbf{S}$: $\texttt{Employee}(\texttt{name})$
- $\mathbf{T}$: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ: $\texttt{Employee}(x) \to \exists y\, \texttt{Dept}(x, y)$

Solutions for $I = \{\texttt{Employee}(\text{Peter})\}$:

The notion of solution: Example

Example

- **S**: $\texttt{Employee}(\texttt{name})$
- **T**: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ: $\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y)$

Solutions for $I = \{\texttt{Employee}(\text{Peter})\}$:

J_1: $\{\texttt{Dept}(\text{Peter}, 1)\}$

The notion of solution: Example

Example

- **S**: $\texttt{Employee}(\texttt{name})$
- **T**: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ: $\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y)$

Solutions for $I = \{\texttt{Employee}(\text{Peter})\}$:

J_1: $\{\texttt{Dept}(\text{Peter},1)\}$

J_2: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{Peter},2)\}$

The notion of solution: Example

Example

- **S**: $\texttt{Employee}(\texttt{name})$
- **T**: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ: $\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y)$

Solutions for $I = \{\texttt{Employee}(\text{Peter})\}$:

J_1: $\{\texttt{Dept}(\text{Peter},1)\}$

J_2: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{Peter},2)\}$

J_3: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{John},1)\}$

The notion of solution: Example

Example

- $\mathbf{S}$: $\texttt{Employee}(\texttt{name})$
- $\mathbf{T}$: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ: $\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y)$

Solutions for $I = \{\texttt{Employee}(\text{Peter})\}$:

J_1: $\{\texttt{Dept}(\text{Peter},1)\}$

J_2: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{Peter},2)\}$

J_3: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{John},1)\}$

J_4: $\{\texttt{Dept}(\text{Peter},n_1)\}$

The notion of solution: Example

Example

- **S**: $\texttt{Employee}(\texttt{name})$
- **T**: $\texttt{Dept}(\texttt{name}, \texttt{number})$
- Σ: $\texttt{Employee}(x) \rightarrow \exists y\, \texttt{Dept}(x, y)$

Solutions for $I = \{\texttt{Employee}(\text{Peter})\}$:

J_1: $\{\texttt{Dept}(\text{Peter},1)\}$

J_2: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{Peter},2)\}$

J_3: $\{\texttt{Dept}(\text{Peter},1), \texttt{Dept}(\text{John},1)\}$

J_4: $\{\texttt{Dept}(\text{Peter},n_1)\}$

J_5: $\{\texttt{Dept}(\text{Peter},n_1), \texttt{Dept}(\text{Peter},n_2)\}$

Canonical universal solution

Algorithm

Input : $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ 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 $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ in Σ **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:

$$\texttt{Employee}(x) \quad \rightarrow \quad \exists y\, \texttt{Dept}(x, y)$$

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

- For $a = \text{Peter}$ do
  - Create a fresh null value n_1
  - Insert $\texttt{Dept}(\text{Peter}, n_1)$ into $J^\star$
- For $a = \text{John}$ do
  - Create a fresh null value n_2
  - Insert $\texttt{Dept}(\text{John}, n_2)$ into $J^\star$

Result: $J^\star = \{\texttt{Dept}(\text{Peter}, n_1), \texttt{Dept}(\text{John}, n_2)\}$

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)

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

Certain answers: Example

Example

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

$$\texttt{Employee}(x) \quad \rightarrow \quad \exists y\, \texttt{Dept}(x, y)$$

Given instance $I = \{\texttt{Employee}(\text{Peter})\}$:

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

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 $J^\star$:

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

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 $J^\star$:

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

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: First part

- The data exchange problem
  - Some fundamental results in relational data exchange
- The need for a more general data exchange framework
  - Two important scenarios: Incomplete databases and knowledge bases

Ongoing project: Reusing schema mappings

$$\mathbf{S} \xrightarrow[\Sigma_{\mathbf{ST}}]{} \mathbf{T}$$

Ongoing project: Reusing schema mappings

$$\mathbf{S} \xrightarrow{\Sigma_{\mathbf{ST}}} \mathbf{T} \xrightarrow{\Sigma_{\mathbf{TU}}} \mathbf{U}$$

Ongoing project: Reusing schema mappings

$$\begin{tikzcd} \mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[rr, bend right, "\Sigma_{\mathbf{SU}}\,?"'] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U} \end{tikzcd}$$

Ongoing project: Reusing schema mappings

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[rr, bend right, "\Sigma_{\mathbf{SU}}\,?"'] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U}
\end{tikzcd}
$$

We need some operators for schema mappings

Ongoing project: Reusing schema mappings

$$\begin{tikzcd} \mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[rr, bend right, "\Sigma_{\mathbf{SU}} = \Sigma_{\mathbf{ST}} \circ \Sigma_{\mathbf{TU}}"'] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U} \end{tikzcd}$$

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?

An inverse operator is also needed

$$\mathbf{S} \xrightarrow[\Sigma_{\mathbf{ST}}]{\qquad\qquad} \mathbf{T}$$

An inverse operator is also needed

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\
\mathbf{V} &
\end{tikzcd}
$$

An inverse operator is also needed

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\
\mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}}?"]
\end{tikzcd}
$$

An inverse operator is also needed

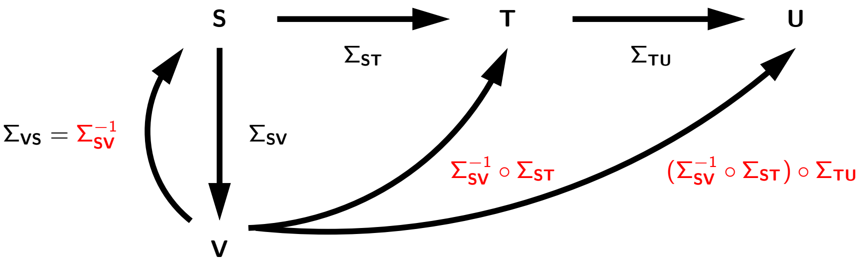
$$\begin{tikzcd}[row sep=large, column sep=huge] \mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\ \mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}} = \color{red}{\Sigma_{\mathbf{SV}}^{-1}}"] \end{tikzcd}$$

An inverse operator is also needed

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\
\mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}} = \Sigma_{\mathbf{SV}}^{-1}"] \arrow[ur, bend right, "\Sigma_{\mathbf{SV}}^{-1} \circ \Sigma_{\mathbf{ST}}"'] &
\end{tikzcd}
$$

Composition and inverse operators have to be combined

An inverse operator is also needed



Composition and inverse operators have to be combined

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.

But this is not the only reason . . .

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 this is not the only reason . . .

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 this is not the only reason . . .

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.

Knowledge exchange: A more general data exchange framework is needed

Example

Assume given the following source knowledge base:

Data:

Father	
Andy	Bob
Bob	Danny
Danny	Eddie

Mother	
Carrie	Bob

Rules:

$$
\begin{aligned}
\texttt{Father}(x, y) &\rightarrow \texttt{Parent}(x, y)\\
\texttt{Mother}(x, y) &\rightarrow \texttt{Parent}(x, y)\\
\texttt{Parent}(x, y) \wedge \texttt{Parent}(y, z) &\rightarrow \texttt{Grandparent}(x, z)
\end{aligned}
$$

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Given a mapping:

$$\begin{aligned} \text{Father}(x, y) &\rightarrow \text{Padre}(x, y) \\ \text{Grandparent}(x, y) &\rightarrow \text{Abuelo}(x, y) \end{aligned}$$

What is a good translation of the initial knowledge base?

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Given a mapping:

$$\begin{array}{rcl} \text{Father}(x, y) & \rightarrow & \text{Padre}(x, y) \\ \text{Grandparent}(x, y) & \rightarrow & \text{Abuelo}(x, y) \end{array}$$

What is a good translation of the initial knowledge base?

Data:

Padre	
Andy	Bob
Bob	Danny
Danny	Eddie

Abuelo	
Andy	Danny
Carrie	Danny
Bob	Eddie

Rules: $\emptyset$

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\begin{aligned} \texttt{Father}(x, y) &\quad\rightarrow\quad \texttt{Parent}(x, y) \\ \texttt{Mother}(x, y) &\quad\rightarrow\quad \texttt{Parent}(x, y) \\ \texttt{Parent}(x, y) \wedge \texttt{Parent}(y, z) &\quad\rightarrow\quad \texttt{Grandparent}(x, z) \end{aligned}$$

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$
\begin{aligned}
\mathtt{Mother}(x, y) &\rightarrow \mathtt{Parent}(x, y) \\
\mathtt{Parent}(x, y) \wedge \mathtt{Parent}(y, z) &\rightarrow \mathtt{Grandparent}(x, z)
\end{aligned}
$$

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\texttt{Parent}(x, y) \wedge \texttt{Parent}(y, z) \quad \rightarrow \quad \texttt{Grandparent}(x, z)$$

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\texttt{Padre}(x, y) \wedge \texttt{Padre}(y, z) \quad \rightarrow \quad \texttt{Abuelo}(x, z)$$

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\texttt{Padre}(x, y) \wedge \texttt{Padre}(y, z) \quad \rightarrow \quad \texttt{Abuelo}(x, z)$$

What data should we materialize?

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\texttt{Padre}(x, y) \wedge \texttt{Padre}(y, z) \quad \rightarrow \quad \texttt{Abuelo}(x, z)$$

What data should we materialize?

Padre	
Andy	Bob
Bob	Danny
Danny	Eddie

Abuelo	
Andy	Danny
Carrie	Danny
Bob	Eddie

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\texttt{Padre}(x, y) \wedge \texttt{Padre}(y, z) \quad \rightarrow \quad \texttt{Abuelo}(x, z)$$

What data should we materialize?

Padre	
Andy	Bob
Bob	Danny
Danny	Eddie

Abuelo	
Carrie	Danny

Knowledge exchange: A more general data exchange framework is needed

Example (cont'd)

Our first alternative does not include any translation of the source rules:

$$\texttt{Padre}(x, y) \wedge \texttt{Padre}(y, z) \quad \rightarrow \quad \texttt{Abuelo}(x, z)$$

What data should we materialize?

Padre	
Andy	Bob
Bob	Danny
Danny	Eddie

Abuelo	
Carrie	Danny

Is this a good translation? Why?

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: Second part

- Formalism for exchanging representations systems
- Applications to incomplete instances
- Applications to knowledge bases
- Concluding remarks

Outline: Second part

- Formalism for exchanging representations systems
- Applications to incomplete instances
- Applications to knowledge bases
- Concluding remarks

Representation systems

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

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

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

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

Representation systems

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

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

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

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

Incomplete instances and knowledge bases are representation systems

In classical data exchange we consider only *complete* data

Recall that given $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, $I \in \text{Inst}(\mathbf{S})$ and $J \in \text{Inst}(\mathbf{T})$: J is a solution for I under $\mathcal{M}$ if $(I, J) \models \Sigma$

$$J \in \text{Sol}_{\mathcal{M}}(I)$$

In classical data exchange we consider only *complete* data

Recall that given $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, $I \in \mathsf{Inst}(\mathbf{S})$ and $J \in \mathsf{Inst}(\mathbf{T})$: J is a solution for I under $\mathcal{M}$ if $(I, J) \models \Sigma$

$$J \in \mathsf{Sol}_{\mathcal{M}}(I)$$

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

$$\mathsf{Sol}_{\mathcal{M}}(\mathcal{X}) \quad = \quad \bigcup_{I \in \mathcal{X}} \mathsf{Sol}_{\mathcal{M}}(I)$$

Extending the definition to representation systems

Given:

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

Extending the definition to representation systems

Given:

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

Definition (APR11)

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

$$\text{rep}(\mathcal{V}) \quad \subseteq \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Extending the definition to representation systems

Given:

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

Definition (APR11)

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

$$\text{rep}(\mathcal{V}) \quad \subseteq \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Or equivalently: $\mathcal{V}$ is an $\mathcal{R}$-*solution* of $\mathcal{U}$ if for every $J \in \text{rep}(\mathcal{V})$, there exists $I \in \text{rep}(\mathcal{U})$ such that $J \in \text{Sol}_{\mathcal{M}}(I)$.

Universal solutions

What is a good solution in this framework?

Universal solutions

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

$$\text{rep}(\mathcal{V}) \quad = \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Strong representation systems

Let $\mathcal{C}$ be a class of mappings.

Strong representation systems

Let $\mathcal{C}$ be a class of mappings.

Definition (APR11)

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

$$\text{rep}(\mathcal{V}) \quad = \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Strong representation systems

Let $\mathcal{C}$ be a class of mappings.

Definition (APR11)

$\mathcal{R} = (\mathbf{W}, \text{rep})$ 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 $\mathcal{U} \in \mathbf{W}$, there exists a $\mathcal{V} \in \mathbf{W}$:

$$\text{rep}(\mathcal{V}) \quad = \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Strong representation systems

Let $\mathcal{C}$ be a class of mappings.

Definition (APR11)

$\mathcal{R} = (\mathbf{W}, \text{rep})$ 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 $\mathcal{U} \in \mathbf{W}$ of type $\mathbf{S}$, there exists a $\mathcal{V} \in \mathbf{W}$:

$$\text{rep}(\mathcal{V}) \quad = \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Strong representation systems

Let $\mathcal{C}$ be a class of mappings.

Definition (APR11)

$\mathcal{R} = (\mathbf{W}, \text{rep})$ 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 $\mathcal{U} \in \mathbf{W}$ of type $\mathbf{S}$, there exists a $\mathcal{V} \in \mathbf{W}$ of type $\mathbf{T}$:

$$\text{rep}(\mathcal{V}) \quad = \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

Strong representation systems

Let $\mathcal{C}$ be a class of mappings.

Definition (APR11)

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

$$\text{rep}(\mathcal{V}) \quad = \quad \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$$

If $\mathcal{R} = (\mathbf{W}, \text{rep})$ is a strong representation system, then the universal solutions for the representatives in **W** can be represented in the same system.

Outline: Second part

- Formalism for exchanging representations systems
- Applications to incomplete instances
- Applications to knowledge bases
- 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
- Computing certain answers

Naive instances

We have already considered naive instances: Instances with null values

- Example: Canonical universal solution

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: Canonical universal solution

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

$$\begin{array}{c} R(1, n_1) \\ R(n_1, 2) \\ R(1, n_2) \end{array}$$

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

$$\text{rep}(\mathcal{I}) \quad = \quad \{K \mid \mu(\mathcal{I}) \subseteq K \text{ for some valuation } \mu\}$$

Naive instances have been extensively used in data exchange:

Proposition (FKMP03)

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

$$rep(\mathcal{J}) \;=\; \mathrm{Sol}_{\mathcal{M}}(I)$$

In fact, the canonical 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:

$$\begin{aligned}\texttt{Manager}(x, y) &\rightarrow \texttt{Reports}(x, y)\\ \texttt{Manager}(x, x) &\rightarrow \texttt{SelfManager}(x)\end{aligned}$$

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

$$\mathcal{J} = \{\texttt{Reports}(n, \text{Peter})\}$$

But $\mathcal{J}$ is not a *good* solution for $\mathcal{I}$.

- It cannot represent the fact that if n is given value Peter, then $\texttt{SelfManager}(\text{Peter})$ 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

Conditional instances

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

$$\begin{array}{l|l} R(1, n_1) & n_1 = n_2 \\ R(n_1, n_2) & n_1 \neq n_2 \ \vee \ n_2 = 2 \end{array}$$

Conditional instances

Example

$$\begin{array}{l|l} R(1, n_1) & n_1 = n_2 \\ R(n_1, n_2) & n_1 \neq n_2 \ \vee \ n_2 = 2 \end{array}$$

Semantics:

Conditional instances

Example

$$\begin{array}{l|l} R(1, n_1) & n_1 = n_2 \\ R(n_1, n_2) & n_1 \neq n_2 \;\vee\; n_2 = 2 \end{array}$$

Semantics:

$\underline{\mu(n_1) = \mu(n_2) = 2}$ $\quad$ $\underline{\mu(n_1) = \mu(n_2) = 3}$ $\quad$ $\underline{\mu(n_1) = 2, \mu(n_2) = 3}$

Conditional instances

Example

$$\begin{array}{l|l} R(1, n_1) & n_1 = n_2 \\ R(n_1, n_2) & n_1 \neq n_2 \ \vee \ n_2 = 2 \end{array}$$

Semantics:

$\mu(n_1) = \mu(n_2) = 2$	$\mu(n_1) = \mu(n_2) = 3$	$\mu(n_1) = 2, \mu(n_2) = 3$
$R(1, 2)$		
$R(2, 2)$		

Conditional instances

Example

$$\begin{array}{l|l} R(1, n_1) & n_1 = n_2 \\ R(n_1, n_2) & n_1 \neq n_2 \;\vee\; n_2 = 2 \end{array}$$

Semantics:

$$\frac{\mu(n_1) = \mu(n_2) = 2}{\begin{array}{c} R(1,2) \\ R(2,2) \end{array}} \qquad \frac{\mu(n_1) = \mu(n_2) = 3}{R(1,3)} \qquad \frac{\mu(n_1) = 2, \mu(n_2) = 3}{}$$

Conditional instances

Example

$$
\begin{array}{l|l}
R(1, n_1) & n_1 = n_2 \\
R(n_1, n_2) & n_1 \neq n_2 \ \vee \ n_2 = 2
\end{array}
$$

Semantics:

$$
\frac{\mu(n_1) = \mu(n_2) = 2}{\begin{array}{c} R(1,2) \\ R(2,2) \end{array}} \qquad \frac{\mu(n_1) = \mu(n_2) = 3}{R(1,3)} \qquad \frac{\mu(n_1) = 2, \mu(n_2) = 3}{\begin{array}{c} \\ R(2,3) \end{array}}
$$

Conditional instances

Example

$$\begin{array}{l|l} R(1, n_1) & n_1 = n_2 \\ R(n_1, n_2) & n_1 \neq n_2 \;\vee\; n_2 = 2 \end{array}$$

Semantics:

$$\frac{\mu(n_1) = \mu(n_2) = 2}{\begin{array}{c} R(1,2) \\ R(2,2) \end{array}} \qquad \frac{\mu(n_1) = \mu(n_2) = 3}{R(1,3)} \qquad \frac{\mu(n_1) = 2, \mu(n_2) = 3}{\begin{array}{c} \\ R(2,3) \end{array}}$$

Interpretations of a conditional instance $\mathcal{I}$:

$$\text{rep}(\mathcal{I}) \quad = \quad \{K \mid \mu(\mathcal{I}) \subseteq K \text{ for some valuation } \mu\}$$

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:

$$\begin{array}{rcl} \texttt{Manager}(x, y) & \rightarrow & \texttt{Reports}(x, y) \\ \texttt{Manager}(x, x) & \rightarrow & \texttt{SelfManager}(x) \end{array}$$

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

$$\begin{array}{l|l} \texttt{Reports}(n, \text{Peter}) & \mathit{true} \\ \texttt{SelfManager}(\text{Peter}) & n = \text{Peter} \end{array}$$

Positive conditional instances are *exactly* the needed representation system

Positive conditional instances are *minimal*:

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*
- *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 Σ is a set of st-tgds.

Positive conditional instance can be used in practice!

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

Theorem (APR11)

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

Positive conditional instance can be used in practice!

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

Theorem (APR11)

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

Let Q be a union of conjunctive queries over $\mathbf{T}$.

$$Q(\mathcal{J}) = \bigcap_{J \in \mathrm{rep}(\mathcal{J})} Q(J)$$

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

Positive conditional instance can be used in practice!

Theorem (APR11)

There exists a polynomial time algorithm that, given a positive conditional instance $\mathcal{I}$ over $\mathbf{S}$, *computes* $\text{certain}_{\mathcal{M}}(Q, \mathcal{I})$.

Positive conditional instance can be used in practice!

Theorem (APR11)

There exists a polynomial time algorithm that, given a positive conditional instance $\mathcal{I}$ *over* $\mathbf{S}$, *computes* $\text{certain}_{\mathcal{M}}(Q, \mathcal{I})$.

The same result holds for the class of unions of conjunctive queries with at most one inequality per disjunct.

- The other important class of queries in the data exchange area for which certain answers can be computed in polynomial time

Outline: Second part

- Formalism for exchanging representations systems
- Applications to incomplete instances
- Applications to knowledge bases
- Concluding remarks

The semantics of *knowledge bases* is given by sets of instances

Knowledge base over $\mathbf{S}$: (I, Γ) such that

- $I \in \text{Inst}(\mathbf{S})$
- Γ a set of rules over $\mathbf{S}$

Semantics: finite models

$$\text{Mod}(I, \Gamma) \quad = \quad \{K \in \text{Inst}(\mathbf{S}) \mid I \subseteq K \text{ and } K \models \Gamma\}$$

We can apply our formalism to knowledge bases

(I_2, Γ_2) is a *KB-solution* for (I_1, Γ_1) under $\mathcal{M}$ if:

$$\mathrm{Mod}(I_2, \Gamma_2) \ \subseteq \ \mathrm{Sol}_{\mathcal{M}}(\mathrm{Mod}(I_1, \Gamma_1))$$

(I_2, Γ_2) is a *universal KB-solution* for (I_1, Γ_1) under $\mathcal{M}$ if:

$$\mathrm{Mod}(I_2, \Gamma_2) \ = \ \mathrm{Sol}_{\mathcal{M}}(\mathrm{Mod}(I_1, \Gamma_1))$$

Motivating questions

Same as for the case of instances with incomplete information.

- Constructing universal KB-solutions
- Answering target queries

New fundamental problem: Construct solutions including as much implicit knowledge as possible.

What are good knowledge-base solutions?

First alternative: universal KB-solutions

But there exist some other KB-solutions desirable to materialize

- Minimality comes into play

What are good knowledge-base solutions?

First alternative: universal KB-solutions

But there exist some other KB-solutions desirable to materialize

- Minimality comes into play

Given sets $\mathcal{X}$, $\mathcal{Y}$ of instances:

- $\mathcal{X} \equiv_{\text{min}} \mathcal{Y}$ if $\mathcal{X}$ and $\mathcal{Y}$ coincide in the minimal instances under $\subseteq$

Definition

(I_2, Γ_2) is a *minimal KB-solution* of (I_1, Γ_1) under $\mathcal{M}$ if:

$$\text{Mod}(I_2, \Gamma_2) \quad \equiv_{\text{min}} \quad \text{Sol}_{\mathcal{M}}(\text{Mod}(I_1, \Gamma_1))$$

Two requirements to construct minimal knowledge-base solutions

Given (I_1, Γ_1) and $\mathcal{M}$, when constructing a minimal KB-solution (I_2, Γ_2) we would like:

Two requirements to construct minimal knowledge-base solutions

Given (I_1, Γ_1) and $\mathcal{M}$, when constructing a minimal KB-solution (I_2, Γ_2) we would like:

1. Γ_2 to only depend on Γ_1 and $\mathcal{M}$:

   Γ_2 is *safe* for Γ_1 and $\mathcal{M}$

Two requirements to construct minimal knowledge-base solutions

Given (I_1, Γ_1) and $\mathcal{M}$, when constructing a minimal KB-solution (I_2, Γ_2) we would like:

1. Γ_2 to only depend on Γ_1 and $\mathcal{M}$:

   Γ_2 is *safe* for Γ_1 and $\mathcal{M}$

Definition

Γ_2 is safe for Γ_1 and $\mathcal{M}$, if for every I_1 there exists I_2:

$$(I_2, \Gamma_2) \text{ is a minimal KB-solution of } (I_1, \Gamma_1) \text{ under } \mathcal{M}$$

Two requirements to construct minimal knowledge-base solutions

2. Γ_2 to be as informative as possible (thus minimizing the size of I_2):

Two requirements to construct minimal knowledge-base solutions

2. Γ_2 to be as informative as possible (thus minimizing the size of I_2):

Definition

Γ_2 is *optimal-safe* if for every other safe set Γ':

$$\Gamma_2 \models \Gamma'$$

Computing minimal KB-solutions

To obtain algorithms for computing minimal KB-solutions, we need to specify the language used in knowledge bases.

- Full st-tgd:

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

Computing minimal KB-solutions

To obtain algorithms for computing minimal KB-solutions, we need to specify the language used in knowledge bases.

- Full st-tgd:

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

Theorem (APR11)

There exists a polynomial-time algorithm that, given $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of full st-tgds, and given a set Γ_1 of full tgds over $\mathbf{S}$, computes a set Γ_2 of second-order logic sentences over $\mathbf{T}$ that is optimal-safe for Γ_1 and $\mathcal{M}$.

Computing minimal KB-solutions

Unfortunately, first-order logic is no expressive enough.

Theorem (APR11)

There exist $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$*, where* Σ *is a set of full st-tgds, and a set* Γ_1 *of full tgds over* $\mathbf{S}$ *such that:*

no FO-sentence is optimal-safe for Γ_1 *and* $\mathcal{M}$*.*

Computing minimal KB-solutions

Unfortunately, first-order logic is no expressive enough.

Theorem (APR11)

There exist $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of full st-tgds, and a set Γ_1 of full tgds over $\mathbf{S}$ such that:

no FO-sentence is optimal-safe for Γ_1 and $\mathcal{M}$.

How can we deal with these problems in practice?

Computing minimal KB-solutions

Unfortunately, first-order logic is no expressive enough.

Theorem (APR11)

There exist $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of full st-tgds, and a set Γ_1 of full tgds over $\mathbf{S}$ such that:

no FO-sentence is optimal-safe for Γ_1 and $\mathcal{M}$.

How can we deal with these problems in practice?

- We need to restrict the language used to specify knowledge bases: Description logics!

Outline: Second part

- Formalism for exchanging representations systems
- Applications to incomplete instances
- Applications to knowledge bases
- Concluding remarks

We can exchange more than complete data

We propose a general formalism to exchange *representation systems*

- Applications to incomplete instances
- 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)

Lots of interesting problems to solve if knowledge bases are specified by means of description logics.

- Better results can be obtained

We can exchange more than complete data

We propose a general formalism to exchange *representation systems*

- Applications to incomplete instances
- 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)

Lots of interesting problems to solve if knowledge bases are specified by means of description logics.

- Better results can be obtained

We can exchange more than complete data

We propose a general formalism to exchange *representation systems*

- Applications to incomplete instances
- 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)

Lots of interesting problems to solve if knowledge bases are specified by means of description logics.

- Better results can be obtained

Thank you!

Bibliography

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

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

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