Foundations of Data Exchange and Metadata Management

Marcelo Arenas
Ron Fagin Special Event - SIGMOD/PODS 2016
The need for a formal definition

- We had a paper with Ron in PODS 2004
- Back then I was a Ph.D. student, and asked Ron whether I could do an internship in IBM Almaden
- He was very positive about the idea, but there were some funding issues
The need for a formal definition

- The solution: applied as Hispanic
- The issue: How Hispanic I am?
  - Is there a precise definition of the notion of being Hispanic?
- The final solution: The IBM Ph.D. fellowship
The old data exchange problem

- The first systems for restructuring and translating data were built several decades ago
  - EXPRESS (1977): A data extraction, processing, and restructuring system

- This problem is particularly relevant today
  - There is a need for a simple, yet general, solution to it
The data exchange problem

How do we specify the relationship between source and target data?

What is a good (declarative) language for this?

How do we materialize a target instance?

What is a good materialization?

Can we do this materialization efficiently?
The data exchange problem

Worker(name)

Emp(name)

Worker(x) → Emp(x)

What is a good materialization?

What do we allow in this rule language?

name
Ron
John

name
Ron
John
Paul
Ringo
The data exchange problem

Worker(name, salary)
Emp(name, dept)

<table>
<thead>
<tr>
<th>name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>100K</td>
</tr>
<tr>
<td>John</td>
<td>90K</td>
</tr>
<tr>
<td>Paul</td>
<td>70K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>?</td>
</tr>
<tr>
<td>John</td>
<td>?</td>
</tr>
<tr>
<td>Paul</td>
<td>?</td>
</tr>
</tbody>
</table>
A solution to the problem

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, Lucian Popa. Data Exchange: Semantics and Query Answering. ICDT 2003

This article proposed a simple, elegant and general solution
- It has a big impact (1000+ citations in Google scholar)
A mapping language

Given: source schema $S$ and a target schema $T$ with no relation names in common

- A source-to-target tuple-generating dependency (st-tgd) is a formula of the form:

\[ \forall x \forall y \varphi(x,y) \rightarrow \exists z \psi(x,z) \]

where $\varphi(x,y)$ and $\psi(x,z)$ are conjunctions of atoms over $S$ and $T$, respectively
A mapping language

A mapping from \( S \) to \( T \) is specified by a set \( \Sigma_{ST} \) of st-tgds.

\[
\begin{align*}
S &= \{ \text{Worker}(\cdot) \} \\
T &= \{ \text{Emp}(\cdot) \} \\
\Sigma_{ST} &= \{ \forall x \text{ Worker}(x) \rightarrow \text{Emp}(x) \} \\
S &= \{ \text{Worker}(\cdot,\cdot) \} \\
T &= \{ \text{Emp}(\cdot,\cdot) \} \\
\Sigma_{ST} &= \{ \forall x \forall y \text{ Worker}(x,y) \rightarrow \exists z \text{ Emp}(x, z) \}
\end{align*}
\]
A definition of a mapping

- A mapping $\mathcal{M}$ is just a tuple $(\mathcal{S}, \mathcal{T}, \Sigma_{ST})$
  - An instance of $\mathcal{S}$ is called a source instance, while an instance of $\mathcal{T}$ is called a target instance
  - $\Sigma_{ST}$ specifies the relationship between source and target data

- What is the semantics of a mapping?
  - When is a target instance considered to be a valid materialization for a source instance under $\mathcal{M}$?
A semantics for mappings

A target instance $J$ is a solution for a source instance $I$ under a mapping $M = (S, T, \Sigma_{ST})$ if:

$$(I,J) \text{ satisfies } \Sigma_{ST} \text{ under the usual semantics of first-order logic}$$
A semantics for mappings

Assume we have a mapping specified by \( \text{Worker}(x) \rightarrow \text{Emp}(x) \) and instances:

\[
I = \{ \text{Worker}(\text{Ron}), \text{Worker}(\text{John}), \text{Worker}(\text{Paul}) \}
\]

\[
J_1 = \{ \text{Emp}(\text{Ron}), \text{Emp}(\text{John}), \text{Emp}(\text{Paul}) \}
\]

\[
J_2 = \{ \text{Emp}(\text{Ron}), \text{Emp}(\text{John}) \}
\]

\(J_1\) is a solution for \(I\): \((I, J_1) \models \forall x \text{ Worker}(x) \rightarrow \text{Emp}(x)\)

\(J_2\) is not a solution for \(I\): and \((I, J_2) \not\models \forall x \text{ Worker}(x) \rightarrow \text{Emp}(x)\)
A semantics for mappings

Consider a mapping specified by $\text{Worker}(x,y) \rightarrow \exists z \text{ Emp}(x, z)$

<table>
<thead>
<tr>
<th>Worker</th>
<th>Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>D1</td>
</tr>
<tr>
<td>John</td>
<td>D2</td>
</tr>
<tr>
<td>Paul</td>
<td>D3</td>
</tr>
<tr>
<td>Ringo</td>
<td>D1</td>
</tr>
</tbody>
</table>
What is a good solution?

The classical notions of null value and homomorphism are used to solve this issue.

- Target instances are allowed to contain constants and nulls.
- Homomorphisms are used to define a notion of most general solution.
Solutions with null values

Consider a mapping specified by $\text{Worker}(x,y) \rightarrow \exists z \text{ Emp}(x,z)$

<table>
<thead>
<tr>
<th>Worker</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>100K</td>
</tr>
<tr>
<td>John</td>
<td>90K</td>
</tr>
<tr>
<td>Paul</td>
<td>70K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>$\perp_1$</td>
</tr>
<tr>
<td>John</td>
<td>$\perp_2$</td>
</tr>
<tr>
<td>Paul</td>
<td>$\perp_3$</td>
</tr>
</tbody>
</table>
The notion of homomorphism

Consider two instances $J_1$ and $J_2$ of the same schema

Consider a function $h$ from the set of constants and nulls to the set of constants and nulls

$h$ is a homomorphism from $J_1$ to $J_2$ if

- $h(c) = c$ for every constant $c$
- if $R(a_1, ..., a_n)$ is a fact in $J_1$, then $R(h(a_1), ..., h(a_n))$ is a fact in $J_2$
The notion of homomorphism

<table>
<thead>
<tr>
<th>Emp</th>
<th>h(Ron) = Ron</th>
<th>h(⊥₁) = D₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>h(John) = John</td>
<td>h(⊥₂) = ⊥₄</td>
</tr>
<tr>
<td>John</td>
<td>h(Paul) = Paul</td>
<td>h(⊥₃) = D₁</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>D₁</td>
</tr>
<tr>
<td>John</td>
<td>⊥₄</td>
</tr>
<tr>
<td>Ringo</td>
<td>D₂</td>
</tr>
<tr>
<td>Paul</td>
<td>D₁</td>
</tr>
</tbody>
</table>
The notion of universal solution

Given a mapping $M$ and a source instance $I$

A solution $J$ for $I$ under $M$ is a universal solution if:

for every solution $K$ for $I$ under $M$, there exists a homomorphism from $J$ to $K$
The notion of universal solution

Consider a mapping specified by \( \text{Worker}(x,y) \rightarrow \exists z \text{ Emp}(x, z) \)

<table>
<thead>
<tr>
<th>Emp</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>⊥_1</td>
</tr>
<tr>
<td>John</td>
<td>⊥_2</td>
</tr>
<tr>
<td>Paul</td>
<td>⊥_3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>D1</td>
</tr>
<tr>
<td>John</td>
<td>D2</td>
</tr>
<tr>
<td>Paul</td>
<td>D3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>⊥</td>
</tr>
<tr>
<td>John</td>
<td>⊥</td>
</tr>
<tr>
<td>Paul</td>
<td>⊥</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>⊥</td>
</tr>
<tr>
<td>John</td>
<td>⊥</td>
</tr>
<tr>
<td>Paul</td>
<td>⊥</td>
</tr>
<tr>
<td>Ringo</td>
<td>⊥_4</td>
</tr>
</tbody>
</table>
Computing universal solutions efficiently

The last ingredient: a polynomial-time algorithm for computing universal solutions

The well-known notion of chase can be used to compute universal solutions

- Existential variables in st-tgds are replaced by fresh nulls
Computing universal solutions efficiently

Consider a mapping specified by $\text{Worker}(x,y) \rightarrow \exists z \text{ Emp}(x,z)$

<table>
<thead>
<tr>
<th>Worker</th>
<th>Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>$\bot_1$</td>
</tr>
<tr>
<td>John</td>
<td>$\bot_2$</td>
</tr>
<tr>
<td>Paul</td>
<td>$\bot_3$</td>
</tr>
</tbody>
</table>
Lessons learned

- Framework has to be simple
  - Syntax and semantics of mappings are simple

- Framework has to be general enough to be of practical interest
  - Based on realistic assumptions
Lessons learned

- Main notions have to be well formalized
  - It is important to have a precise definition of what a valid translation of a source instance is

- Do not reinvent the wheel: use well-known and widely-studied concepts, bring tools from other areas
  - Syntax and semantics of mappings are based on first-order logic
  - Universal solutions are defined in terms of homomorphisms
And this is not all …

The fundamental problem of answering target queries was also considered.

How do we answer a target query?

<table>
<thead>
<tr>
<th>Worker</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>100K</td>
</tr>
<tr>
<td>John</td>
<td>90K</td>
</tr>
<tr>
<td>Paul</td>
<td>70K</td>
</tr>
</tbody>
</table>
A semantics for target queries

- How to evaluate a query $Q$ over an instance $I$ is well understood
  - $Q(I)$ is used to denote the answer to $Q$ over $I$

- This notion is used to define the answer to a target query with respect to a source instance given a mapping
A semantics for target queries

Given a mapping $M$, a source instance $I$ and a query $Q$ over the target schema

The set of certain answers of $Q$ with respect to $I$ given $M$ is defined as:

$$\text{certain}_M(Q,I) = \bigcap_{J \text{ is a solution for } I \text{ under } M} Q(J)$$
A semantics for target queries

Consider a mapping specified by \( \text{Worker}(x, y) \rightarrow \exists z \text{Emp}(x, z) \)
and the target query \( Q(u) = \exists v \text{Emp}(u, v) \)

<table>
<thead>
<tr>
<th>Worker</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron</td>
<td>100K</td>
</tr>
<tr>
<td>John</td>
<td>90K</td>
</tr>
<tr>
<td>Paul</td>
<td>70K</td>
</tr>
</tbody>
</table>

Answer to \( Q = \{ \text{Ron, John, Paul} \} \)

\( \text{certain}_M(Q, I) = \{ \text{Ron, John, Paul} \} \)

Answer to \( Q = \{ \text{Ron, John, Paul, Ringo} \} \)
Computing certain answers efficiently

Given a mapping $M$, a source instance $I$ and a universal solution $J$ for $I$ under $M$

For every union of conjunctive queries $Q$:

$$\text{certain}_M(Q,I) = \{ a \mid a \in Q(J) \text{ and } a \text{ only mentions constants} \}$$
Why this approach was so influential?

The simple and well-defined framework for data exchange open many directions for further research

- Design of efficient algorithms for computing universal solutions (minimal ones)
- Design of efficient query answering algorithms for target positive queries
- Identification of more expressive query languages (inequalities, negation and aggregation)
- Use of source and target integrity constraints
- Optimization of mappings
Why this approach was so influential?

The simple and well-defined framework for data exchange open many directions for further research:

- Use of more expressive mapping languages
- Study of different notions of solutions and semantics for query answering (OWA versus CWA)
- Development of data exchange settings in other data models: XML, graph databases, probabilistic databases
- Development of mapping operators
Metadata management

\[ M_{13} = M_{12} \circ M_{23} \]
Metadata management

\[ \hat{M}_{14}^{-1} \circ M_{14} \circ M_{12} \circ M_{23} \]
The need for mapping operators

Philip A. Bernstein. Applying Model Management to Classical Meta Data Problems. CIDR 2003

- Once a formal definition of mapping is given, these operators can be formally defined and studied
The composition operator

- $S_1$, $S_2$ and $S_3$ denote pairwise disjoint schemas
  - Instances of $S_k$ are denoted as $I_k$ ($k = 1, 2, 3$)
- $M_{12}$ and $M_{23}$ denote mappings from $S_1$ to $S_2$ and $S_2$ to $S_3$, respectively
The composition operator

The composition of $M_{12}$ with $M_{23}$ is defined as a mapping $M_{12} \circ M_{23}$ such that:

$I_3$ is a solution for $I_1$ under $M_{12} \circ M_{23}$ if and only if there exists $I_2$ such that
$I_2$ is a solution for $I_1$ under $M_{12}$ and
$I_3$ is a solution for $I_2$ under $M_{23}$
A mapping language for the composition operator

- What is the right language to express the composition operator?

- Are st-tgds closed under composition?
- If $M_{12}$ and $M_{23}$ are specified by sets of st-tgds, can also $M_{12} \circ M_{23}$ be specified by a set of st-tgds?
A mapping language for the composition operator

Having a formal definition of mappings these questions can be answered

- st-tgds are not closed under composition
  - There exist mappings $M_{12}$ and $M_{23}$ specified by sets of st-tgds, such that $M_{12} \circ M_{23}$ cannot be specified by a set of st-tgds

- There exists a mapping language that is appropriate for composition
The power of composition

Consider a mapping $M_{12}$ specified by the following st-tgds:

- $\text{Node}(x) \rightarrow \exists u \ \text{Paint}(x, u)$
- $\text{Edge}(x, y) \rightarrow \text{Arc}(x, y)$

and a mapping $M_{23}$ specified by the following st-tgds:

- $\text{Paint}(x, u) \rightarrow \text{Color}(u)$
- $\text{Arc}(x, y) \land \text{Paint}(x, u) \land \text{Paint}(y, u) \rightarrow \text{Error}(x) \land \text{Error}(y)$
Adding second-order quantification

Unless P = NP, the previous mapping $M_{12} \circ M_{23}$ cannot be defined in first-order logic.

What does it need to be added to st-tgds to capture the composition of two mappings?

- Fagin’s theorem gives us a good idea of what needs to be added: $NP = \exists SO$
The language of second-order st-tgds

A simple extension of st-tgds gives rise to a mapping language that is appropriate to define the composition:

$$\exists f \left( \forall x \ [ \text{Node}(x) \rightarrow \text{Color}(f(x)) ] \land 
\forall x \forall x [ \text{Edge}(x,y) \land f(x) = f(y) \rightarrow \text{Error}(x) \land \text{Error}(y) ] \right)$$

These dependencies are called second-order st-tgds (SO tgds)
The language of second-order tgds

It is the right language for specifying the composition of mappings defined by st-tgds

- The composition of a sequence of mappings specified by sets of st-tgds can be specified by an SO tgd

- SO tgds are closed under composition

- For every SO tgd $\varphi$, there exists a sequence of mappings specified by sets of st-tgds such that its composition is specified by $\varphi$
The language of second-order tgds

Besides, it has (almost) the same good properties as st-tgds for data exchange

- Universal solutions are defined in the same way
- There is a polynomial-time algorithm (based on the chase) for computing universal solutions
- Certain answers to union of conjunctive queries can be computed by using universal solutions
Lessons learned

- Having a simple and formal definition of mappings is a key ingredient to study mapping operators.

- Use well-known and widely-studied concepts.
  - A simple form of second-order quantification gives rise to a simple yet powerful mapping language that is appropriate to define the composition operator.
Thanks Ron for many well-defined and inspiring concepts!