Laconic schema mappings: computing the core with SQL queries

Balder ten Cate (INRIA and ENS Cachan)
Laura Chiticariu (IBM Almaden)
Phokion Kolaitis (UC Santa Cruz and IBM Almaden)
Wang-Chiew Tan (UC Santa Cruz)

VLDB '09, Lyon
Data exchange

- **Data exchange**: transforming data structured under a source schema into data structured under a different target schema.
One of the first steps is to specify the relationships between the schemas. This is known to be a difficult task.

Generic architecture of mapping systems
e.g., IBM Clio
HePToX
Altova MapForce
Stylus Studio
MS Biztalk Mapper
BEA Aqualogic
Schema mappings and data exchange

- **Schema mapping** $M=(S,T,\Sigma)$:

  A declarative specification of the relationships between two database schemas (a **source schema** $S$ and a **target schema** $T$), in the form of a list $\Sigma$ of constraints (sentences of some logical language to be specified).

- **Data exchange** (formal definition [FKMP'05]):

  Given a schema mapping and a **source instance** $I$ of the source schema, construct a **solution** of $I$: an instance $J$ of the target schema, such that $(I,J) \models \Sigma$.

- **NB**: Schema mappings are used also in other data-interoperability tasks such as data integration.
The constraints $\Sigma$ are commonly specified by source-to-target tuple generating dependencies (s-t tgds) [FKMP'05]:

- An s-t tgd is a FO-sentence $\forall x (\varphi_S(x) \rightarrow \exists y. \psi_T(x,y))$ with
  - $\varphi_S(x)$ a conjunction of atomic formulas over the source schema.
  - $\psi_T(x,y)$ a conjunction of atomic formulas over the target schema.

Example: $\forall x_1 x_2 (R x_1 x_2 x_3 \rightarrow \exists y. T x_1 x_2 y)$

"Whenever the source instance contains a fact $Rabc$, the target instance should contain $Tabd$ for some value $d".
Which solution?

A source instance may have many solutions. Example:

- Schema mapping: \( \Sigma = \{ \forall x_1 x_2 x_3 (R x_1 x_2 x_3 \rightarrow T x_1 x_2) \} \)
- Source instance: \( I = \{ Rabc \} \)
- Solutions \( J_1 = \{ Tab \} \) and \( J_2 = \{ Tab, Tac \} \)

Definition: a universal solution is a solution that maps homomorphically into every other solution [FKMP'05].

- Intuitively, univ. solutions contain no more information than needed.
- In the above example, \( J_1 \) is universal while \( J_2 \) is not.

Two particular universal solutions: the canonical universal solution and the core universal solution.
Recall that s-t tgds are **logical constraints** expressing relationships between the source schema and target schema.

They also have another, **procedural** reading. For example,

- $\forall x_1, x_2 (R_{x_1 x_2} \rightarrow \exists y. T_{x_1 x_2 y})$

  - **Declarative semantics**: for every fact $R_{ab}$, the target instance needs to contain $T_{abc}$ for some $c$.

  - **Procedural reading**: for every fact $R_{ab}$, choose a fresh null value $N$ and add $T_{abN}$ to the target instance.

The target instance obtained by "executing" the s-t tgds (as instructions) is the **canonical universal solution** [FKMP'05].

This procedure is known as **the chase**.
• **Note**: the "procedural reading" of the s-t tgds (i.e., the chase) is just **one way to compute a universal solution**.

• **Advantage**: the canonical universal solution is easy to compute
  - A schema mapping specified by s-t tgds can be compiled into a list of **SQL queries** computing the canonical universal solution.
  - We can compute the canonical univ. solution using any RDBMS.
  - This is how data exchange systems such as Clio work.

• **Disadvantage**: the canonical universal solution may contain more null values and more facts than strictly necessary.
Core universal solution

- The core universal solution is the smallest universal solution (unique-up-to-isomorphism) [FKP'05]

- Nice properties of the core universal solution:
  - It is the universal solution that contains the fewest null values.
  - It is the universal solution that satisfies the most dependencies.
  - It is the universal solution that gives the most conservative answers to CQ* queries.

- Formally,
  - the core of an instance is smallest homomorphically equivalent instance (unique-up-to-isomorphism).
  - the core universal solution of a source instance is the core of the canonical universal solution (or, of any other universal solution).
Computing the core universal solution

- Computing the core of an arbitrary instance (with nulls) is NP-hard.
- Computing core universal solutions for schema mappings specified by s-t tgds can be done in PTime (data complexity) due to special properties of the canonical universal solution.
- Several algorithms have been proposed:
  - Greedy [FKP'05], Blocks [FKP'05], Gottlob-Nash [GN'08]
- all these algorithms involve recursively performing two steps
  1. testing for the existence of a homomorphism from J into a proper subinstance of J, and
  2. removing facts from J.

which goes beyond SQL, i.e., out-of-the-box data transformation.
Example

Schema mapping

\[ S: \]
- PTStudent(age, name)
- GradStudent(age, name)

\[ T: \]
- Advised(sname, facid)
- WorksWith(sname, facid)

\[ \Sigma: \]
- \( PTStudent(x, y) \rightarrow \exists z. Advised(y, z) \)
- \( GradStudent(x, y) \rightarrow \exists z. (Advised(y, z) \land WorksWith(y, z)) \)

Source instance

<table>
<thead>
<tr>
<th>PTStudent</th>
<th>GradStudent</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>name</td>
</tr>
<tr>
<td>32</td>
<td>John</td>
</tr>
<tr>
<td>30</td>
<td>Ann</td>
</tr>
<tr>
<td>age</td>
<td>name</td>
</tr>
<tr>
<td>27</td>
<td>Bob</td>
</tr>
<tr>
<td>30</td>
<td>Ann</td>
</tr>
</tbody>
</table>

Canonical univ. sol.

<table>
<thead>
<tr>
<th>Advised</th>
<th>WorksWith</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>facid</td>
</tr>
<tr>
<td>John</td>
<td>N1</td>
</tr>
<tr>
<td>Ann</td>
<td>N2</td>
</tr>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
</tbody>
</table>

Core univ. sol.

<table>
<thead>
<tr>
<th>Advised</th>
<th>WorksWith</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>facid</td>
</tr>
<tr>
<td>John</td>
<td>N1</td>
</tr>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
<tr>
<td>sname</td>
<td>facid</td>
</tr>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
</tbody>
</table>
Why core universal solutions are not used in practice

- **Canonical universal solution**: less preferred but easy to compute (can be done using SQL queries).
- **Core universal solution**: most preferred solution but hard to compute (out-of-the-box).
- **Clio** (the IBM Almaden prototype data exchange engine) constructs only **canonical universal solutions**:
  - No need to implement a data transformation engine
  - Leverage optimizations built into the DBMS.
- **Can we compute core universal solution using SQL queries?**
Laconicity: have the cake and eat it

- A schema mapping is laconic if, for every source instance, the canonical universal solution is the core universal solution.

- Our main contribution: an algorithm for transforming every schema mapping specified by s-t tgds into an equivalent laconic schema mapping (assuming a linear order on the source domain).

- Algorithm for computing core universal solution:
  - Transform schema mapping $M$ into equivalent laconic $M'$.
  - Compile $M'$ into SQL queries computing canonical univ. sol.
  - Apply SQL queries to the source instance to obtain core univ. sol.

- Thus, yes we can compute the core universal solution with SQL queries using any RDBMS.
Example revisited

Schema mapping

\[ \Sigma: \]
\begin{align*}
&\text{PTStudent}(x,y) \land \lnot \exists u. \text{GradStudent}(u,y) \rightarrow \exists z. \text{Advised}(y,z) \\
&\text{GradStudent}(x,y) \rightarrow \exists z.(\text{Advised}(y,z) \land \text{WorksWith}(y,z))
\end{align*}

Source instance

<table>
<thead>
<tr>
<th>PTStudent</th>
<th>GradStudent</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>name</td>
</tr>
<tr>
<td>32</td>
<td>John</td>
</tr>
<tr>
<td>30</td>
<td>Ann</td>
</tr>
<tr>
<td>age</td>
<td>name</td>
</tr>
<tr>
<td>27</td>
<td>Bob</td>
</tr>
<tr>
<td>30</td>
<td>Ann</td>
</tr>
</tbody>
</table>

Canonical univ. sol.

<table>
<thead>
<tr>
<th>Advised</th>
<th>WorksWith</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>facid</td>
</tr>
<tr>
<td>John</td>
<td>N1</td>
</tr>
<tr>
<td>Ann</td>
<td>N2</td>
</tr>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
</tbody>
</table>

Core univ.sol.

<table>
<thead>
<tr>
<th>Advised</th>
<th>WorksWith</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>facid</td>
</tr>
<tr>
<td>John</td>
<td>N1</td>
</tr>
<tr>
<td>Bob</td>
<td>N3</td>
</tr>
<tr>
<td>Ann</td>
<td>N4</td>
</tr>
</tbody>
</table>
Idea behind the translation

- For schema mappings given by a finite set of s-t tgds,
  1. each core universal solution can be decomposed into small fact blocks, each having disjoint sets of null values.
  2. there is a finite set of fact-block types that can arise
  3. for each fact-block type $t$, there is an SQL query $precon_t$ that tells exactly when a fact block of type $t$ needs to be created.
- Complications arise with source instances are non-rigid (i.e., have non-trivial automorphisms). These are solved by assuming a linear order on the source domain
Example revisited

Schema mapping

$S$: Colleague(name, name)  

$T$: ReportsTo(name, name)

$\Sigma$: Colleague(x, y) $\rightarrow \exists z$. ReportsTo(x, z) $\land$ ReportsTo(y, z)

Source instance

<table>
<thead>
<tr>
<th>Colleague</th>
<th>Canonical univ. sol.</th>
<th>Core univ. sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>John</td>
<td>Mary</td>
<td>John</td>
</tr>
<tr>
<td>Mary</td>
<td>John</td>
<td>Mary N1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mary N2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>John N2</td>
</tr>
</tbody>
</table>

Has a non-trivial automorphism:

\{ John $\rightarrow$ Mary, Mary $\rightarrow$ John \}

Hence, whenever a null is created for the pair (John, Mary), another null is created for the pair (Mary, John). Can be avoided by requiring that $x \leq y$. 
Optimality of the results

- Our laconisation algorithm is optimal in several ways:
  1. **Exponential blow-up** in size of schema mapping is **unavoidable**.
  2. The use of **negation** in the left-hand sides is **necessary** (in general the left-hand sides will be **Boolean combinations of conjunctive queries**)
  3. The use of a **linear order** on the source domain is **necessary** (unless the right-hand side of each s-t tgd is selfjoin-free)
  4. Our method **cannot be extended with target constraints**.
Summary and outlook

- In data exchange there is a choice which solution to materialize: canonical universal solution vs core universal solution.
- Laconic schema mappings: have the cake and eat it
  - the **canonical universal solution** is the **core universal solution**
- Our main contributions:
  1. an algorithm to transform schema mappings given by s-t tgds into equivalent laconic schema mappings (given by FO s-t tgds)
  2. a set of results showing that the translation is optimal
- Future work: **implementation** and **empirical evaluation**.
- **Note**: In [Mecca, Papotti, and Raunich SIGMOD'09], similar results were obtained independently for a restricted class of s-t tgds and used to obtain considerable performance gains.