Composing Schema Mapping
An Overview

Phokion G. Kolaitis

UC Santa Cruz & IBM Research – Almaden

Joint work with
R. Fagin, L. Popa, and W.C. Tan
Data Interoperability

- Data may reside
  - at several different sites
  - in several different formats (relational, XML, ...).

- Applications need to access, process, and query these data.

- **Data Exchange:**
  - A fundamental problem in data interoperability
  - Described as the "oldest problem in databases"
  - Formalized and studied in depth in the past 15 years.
Data Exchange

Transform data structured under a source schema into data structured under a different target schema.
Schema Mappings

• **Schema mappings:**
High-level, declarative assertions that specify the relationship between two database schemas.

• Schema mappings constitute the essential **building blocks** in formalizing and studying data interoperability tasks, including **data exchange**.

• Schema mappings make it possible to separate the **design** of the relationship between schemas from its implementation.
Schema Mappings

Schema Mapping $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$

- **Source schema** $\mathbf{S}$, **Target schema** $\mathbf{T}$
- $\Sigma$: High-level, declarative assertions that specify the relationship between $\mathbf{S}$ and $\mathbf{T}$.

**Question:** What is a “good” schema-mapping specification language?
Schema-Mapping Specification Languages

- **Obvious Idea:**
  Use a logic-based language to specify schema mappings. In particular, use first-order logic.

- **Warning:**
  Unrestricted use of first-order logic as a schema-mapping specification language gives rise to *undecidability* of basic algorithmic problems about schema mappings.
Schema Mapping Specification Languages

Let us consider some simple tasks that every schema-mapping specification language should support:

- **Copy (Nicknaming):**
  - Copy each source table to a target table and rename it.

- **Projection:**
  - Form a target table by projecting on one or more columns of a source table.

- **Column Augmentation:**
  - Form a target table by adding one or more columns to a source table.

- **Decomposition:**
  - Decompose a source table into two or more target tables.

- **Join:**
  - Form a target table by joining two or more source tables.

- **Combinations of the above** (e.g., “join + column augmentation + …”)
Schema Mapping Specification Languages

– Copy (Nicknaming):
  • $\forall x_1, \ldots, x_n (P(x_1, \ldots, x_n) \rightarrow R(x_1, \ldots, x_n))$

– Projection:
  • $\forall x,y,z (P(x,y,z) \rightarrow R(x,y))$

– Column Augmentation:
  • $\forall x,y (P(x,y) \rightarrow \exists z R(x,y,z))$

– Decomposition:
  • $\forall x,y,z (P(x,y,z) \rightarrow R(x,y) \land T(y,z))$

– Join:
  • $\forall x,y,z (E(x,z) \land F(z,y) \rightarrow R(x,y,z))$

– Combinations of the above (e.g., “join + column augmentation + …”)
  • $\forall x,y,z (E(x,z) \land F(z,y) \rightarrow \exists w (R(x,y) \land T(x,y,z,w)))$
Schema Mapping Specification Languages

• **Question:** What do all these tasks (copy, projection, column augmentation, decomposition, join) have in common?

• **Answer:**
  They can be specified using
  **GLAV (global-and-local-as-view) constraints,**
  also known as
  **source-to-target tuple generating dependencies (s-t tgds).**
The relationship between source and target is given by GLAV constraints (s-t tgds)

\[ \forall \mathbf{x} \ (\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \ \psi(\mathbf{x}, \mathbf{y})) \text{, where} \]

- \( \varphi(\mathbf{x}) \) is a conjunction of atoms over the source;
- \( \psi(\mathbf{x}, \mathbf{y}) \) is a conjunction of atoms over the target.

- GLAV constraints assert that:

  some conjunctive query over the source is contained in some other conjunctive query over the target.

**Example:**

\[ \forall s \ \forall c \ (\text{Student}(s) \land \text{Enrolls}(s, c)) \rightarrow \exists t \ \exists g \ (\text{Teaches}(t, c) \land \text{Grade}(s, c, g)) \]
Data Exchange via the schema mapping $M = (S, T, \Sigma)$

Given a source instance $I$, construct a target instance $J$, so that $(I, J)$ satisfy the specifications $\Sigma$ of $M$.

Such a $J$ is called a solution for $I$.

Difficulty:
- Usually, there are multiple solutions
- Which one is the “best” to materialize?
Fagin, K …, Miller, Popa:
Identified and studied the concept of a universal solution for GLAV mappings, i.e., schema mappings specified by a finite set of GLAV constraints.

– A universal solutions is a most general solution.
– A universal solution “represents” the entire space of solutions.
– A “canonical” universal solution can be generated efficiently using the chase procedure.
Universal Solutions in Data Exchange

Schema $S$  

Schema $T$

$I$  

$J$

$\Sigma$

Universal Solution

$J_1$

$J_2$

$J_3$

Homomorphisms

Solutions

$h_1$

$h_2$

$h_3$
Managing Schema Mappings

• Schema mappings can be quite complex.

• Methods and tools are needed to automate or semi-automate schema-mapping management.

• Metadata Management Framework – Bernstein 2003 based on generic schema-mapping operators:
  – Match operator
  – Merge operator
  – Composition operator
  – Inverse operator
Composing Schema Mappings

- Given $M_{12} = (S_1, S_2, \Sigma_{12})$ and $M_{23} = (S_2, S_3, \Sigma_{23})$, derive a schema mapping $M_{13} = (S_1, S_3, \Sigma_{13})$ that is "equivalent" to the sequential application of $M_{12}$ and $M_{23}$.
- $M_{13}$ is a composition of $M_{12}$ and $M_{23}$, denoted

$$M_{13} = M_{12} \circ M_{23}$$
Composing Schema Mappings

• Given $M_{12} = (S_1, S_2, \Sigma_{12})$ and $M_{23} = (S_2, S_3, \Sigma_{23})$, derive a schema mapping $M_{13} = (S_1, S_3, \Sigma_{13})$ that is “equivalent” to the sequence $M_{12}$ and $M_{23}$.

What does it mean for $M_{13}$ to be “equivalent” to the composition of $M_{12}$ and $M_{23}$?
Earlier Work

- **Metadata Model Management** (Bernstein in CIDR 2003)
  - Composition is one of the fundamental operators
  - However, no precise semantics is given

- **Composing Mappings among Data Sources** (Madhavan & Halevy in VLDB 2003)
  - First to propose a semantics for composition
  - Their notion of composition depends on the class of queries; it may not be unique up to logical equivalence.
Semantics of Composition

• Every schema mapping \(M = (S, T, \Sigma)\) defines a binary relationship \(\text{Inst}(M)\) between instances:
  \[
  \text{Inst}(M) = \{ (I,J) \mid (I,J) \models \Sigma \}.
  \]
  From a semantic point of view, a schema mapping \(M\) can be identified with the set \(\text{Inst}(M)\).

• **Definition:** (FKPT)
  A schema mapping \(M_{13}\) is a composition of \(M_{12}\) and \(M_{23}\) if
  \[
  \text{Inst}(M_{13}) = \text{Inst}(M_{12}) \circ \text{Inst}(M_{23}), \quad \text{that is,}
  \]
  \[
  (I_1,I_3) \models \Sigma_{13}
  \]
  if and only if
  there exists \(I_2\) such that \((I_1,I_2) \models \Sigma_{12}\) and \((I_2,I_3) \models \Sigma_{23}\).
The Composition of Schema Mappings

**Fact:** If both $M = (S_1, S_3, \Sigma)$ and $M' = (S_1, S_3, \Sigma')$ are compositions of $M_{12}$ and $M_{23}$, then $\Sigma$ are $\Sigma'$ are logically equivalent.

For this reason:

- We say that $M$ (or $M'$) is *the composition* of $M_{12}$ and $M_{23}$.
- We write $M_{12} \circ M_{23}$ to denote it.
Issues in Composition of Schema Mappings

- The semantics of composition was the first main issue.

- The second main issue is the language of the composition.
  
  - Is the language of GLAV constraints closed under composition?
    
    If $M_{12}$ and $M_{23}$ are GLAV mappings, is $M_{12} \circ M_{23}$ a GLAV mapping as well?

  - If not, what is the “right” language for composing schema mappings?
**Inexpressibility of Composition**

**Theorem:**

- GLAV mappings are **not** closed under composition.

- In fact, there are GLAV mappings $M_{12}$ and $M_{23}$ such that their composition $M_{12} \circ M_{23}$ is **not** expressible in least fixed-point logic LFP; in particular, $M_{12} \circ M_{23}$ is **not** expressible in first-order logic FO.
Lower Bounds for Composition

- \( M_{12} \):
  \[ \forall x \forall y (E(x,y) \rightarrow \exists u \exists v (C(x,u) \land C(y,v))) \]
  \[ \forall x \forall y (E(x,y) \rightarrow F(x,y)) \]

- \( M_{23} \):
  \[ \forall x \forall y \forall u \forall v (C(x,u) \land C(y,v) \land F(x,y) \rightarrow D(u,v)) \]

- Given graph \( G=(V, E) \):
  - Let \( I_1 = E \)
  - Let \( I_3 = \{ (r,g), (g,r), (b,r), (r,b), (g,b), (b,g) \} \)

**Fact:**
\( G \) is 3-colorable iff \( <I_1, I_3> \in \text{Inst}(M_{12}) \circ \text{Inst}(M_{23}) \)

- **Theorem (Dawar – 1998):**
  3-Colorability is **not** expressible in LFP.
Complexity of Composition

**Definition:** The model checking problem for a schema mapping \( M = (S, T, \Sigma) \) asks: given a source instanced \( I \) and a target instance \( J \), does \( \langle I, J \rangle \models \Sigma \) ?

**Fact:** If \( M \) is a GLAV mapping, then the model checking problem for \( M \) is in LOGSPACE.

**Fact:** There are GLAV mappings \( M_{12} \) and \( M_{23} \) such that the model checking problem for their composition \( M_{12} \circ M_{23} \) is NP-complete.
Employee Example

\(M_{12} : \forall e (\text{Emp}(e) \rightarrow \exists m \text{ Rep}(e,m))\)

\(M_{23} : \forall e \forall m (\text{Rep}(e,m) \rightarrow \text{Mgr}(e,m))\)

\(\forall e (\text{Rep}(e,e) \rightarrow \text{SelfMgr}(e))\)

**Theorem:**
- The composition \(M_{12} \circ M_{23}\) is **not** definable by any set (finite or infinite) of GLAV constraints.
- The composition \(M_{12} \circ M_{23}\) is definable in a fragment of Second-Order Logic that extends GLAV constraints with Skolem functions.
Employee Example - revisited

**M\textsubscript{12}**: 
- \( \forall e \ (\text{Emp}(e) \rightarrow \exists m \ \text{Rep}(e,m)) \)

**M\textsubscript{23}**: 
- \( \forall e \ \forall m (\text{Rep}(e,m) \rightarrow \text{Mgr}(e,m)) \)
- \( \forall e \ (\text{Rep}(e,e) \rightarrow \text{SelfMgr}(e)) \)

**Fact:** \( M\textsubscript{12} \circ M\textsubscript{23} \) is definable by the following SO-tgd

\[ \exists f \ (\forall e (\text{Emp}(e) \rightarrow \text{Mgr}(e, f(e))) \land \forall e (\text{Emp}(e) \land (e = f(e)) \rightarrow \text{SelfMgr}(e))) \]
Second-Order Tgds

**Definition:** Let \( S \) be a source schema and \( T \) a target schema. A *second-order tuple-generating dependency* (SO tgd) is a formula of the form:

\[
\exists f_1 \ldots \exists f_m ( (\forall x_1(\phi_1 \rightarrow \psi_1)) \land \ldots \land (\forall x_n(\phi_n \rightarrow \psi_n)) ),
\]
where

- Each \( f_i \) is a function symbol.
- Each \( \phi_i \) is a conjunction of atoms from \( S \) and equalities of terms.
- Each \( \psi_i \) is a conjunction of atoms from \( T \).

**Example:**

\[
\exists f (\forall e(Emp(e) \rightarrow Mgr(e,f(e))) \land \\
\forall e(\ Emp(e) \land (e=f(e)) \rightarrow SelfMgr(e)))
\]
Composing SO-Tgds and Data Exchange

**Theorem** (FKPT):

- The composition of two SO-tgds is definable by a SO-tgd.

- There is an algorithm for composing SO-tgds.

- The chase procedure can be extended to SO-tgds; it produces universal solutions in polynomial time.

- Every SO tgd is the composition of finitely many finite sets of s-t tgds. Hence, SO tgds are the “right” language for the composition of s-t tgds.
When is the composition FO-definable?

Fact:

- It is an undecidable problem to tell whether the composition of two GLAV mappings is FO-definable.

- However, there are certain sufficient conditions that guarantee that the composition of two GLAV mappings is FO-definable.
LAV, Extended LAV, and GAV Mappings

**GLAV constraints:** \( \forall x (\varphi(x) \rightarrow \exists y \psi(x,y)) \),

- **LAV (local-as-view) constraints:**
  \( \forall x (P(x) \rightarrow \psi(x)) \), where each variable occurs *only once* in \( P(x) \).
  - Copy, Projection, Column Augmentation, Decomposition, …

- **Extended LAV constraints:**
  \( \forall x (P(x) \rightarrow \psi(x)) \), where \( P \) is a source relation
  (a variable may occur more than once in \( P(x) \))
  - \( \forall e (\text{Rep}(e,e) \rightarrow \text{SelfMgr}(e)) \)

- **GAV (global-as-view) constraints:**
  \( \forall x (\varphi(x) \rightarrow R(x)) \), where \( R \) is a target relation
  - Copy, Projection, Join
Composing GLAV Schema Mappings

<table>
<thead>
<tr>
<th>Composition</th>
<th>Logically Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAV ◦ GAV</td>
<td>GAV</td>
</tr>
<tr>
<td>GAV ◦ GLAV</td>
<td>GLAV</td>
</tr>
<tr>
<td>GLAV ◦ LAV</td>
<td>GLAV</td>
</tr>
<tr>
<td>LAV ◦ Extended LAV</td>
<td>not GLAV</td>
</tr>
<tr>
<td>LAV ◦ GAV</td>
<td>not GLAV</td>
</tr>
</tbody>
</table>

Note:

- **LAV ◦ LAV** equivalent to **GLAV** (special case of **GLAV ◦ LAV**) was established by Arocena, Fuxman, Miller (2010).
- **LAV ◦ Extended LAV** - Employee schema mapping
- **LAV ◦ GAV** - 3-Colorability schema mapping
Synopsis of Schema Mapping Composition

• GLAV mappings are **not** closed under composition.

• SO-tgds form a **well-behaved** fragment of second-order logic.
  
  – SO-tgds are closed under composition; they are the “**right**” language for composing GLAV mappings.
  
  – SO-tgds are “**chasable**”:
    Polynomial-time data exchange with universal solutions.

• SO-tgds and the composition algorithm have been incorporated in the **IBM InfoSphere Information Server**.
Related Work

• Composition with respect to conjunctive-query equivalence
  Madhavan-Halevy – 2003, K … and Fagin – 2012,
  Arenas, Pérez, Reutter - 2013

• Composing more expressive schema mappings
  Nash, Bernstein, Melnik – 2007

• Composing XML Schema Mappings
  Amano, Libkin, Murlak – 2009

• Categorical treatment of data migration
“The notion of composition of maps leads to the most natural account of fundamental notions of mathematics, from multiplication, addition, and exponentiation, through the basic notions of logic.”

"Conceptual Mathematics"
by
W. Lawvere and S. Schanuel