Schema Mappings
Data Exchange
&
Metadata Management

Phokion G. Kolaitis
IBM Almaden Research Center

joint work with

Ronald Fagin   Renée J. Miller   Lucian Popa   Wang-Chiew Tan
IBM Almaden      U. Toronto      IBM Almaden      UC Santa Cruz
The Data Interoperability Problem

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

- Two different, but related, facets of data interoperability:
  - **Data Integration** (aka **Data Federation**):
  - **Data Exchange** (aka **Data Translation**):
Data Integration

Query heterogeneous data in different sources via a virtual global schema
Data Exchange

Transform data structured under a source schema into data structured under a different target schema.
Data Exchange

Data Exchange is an old, but recurrent, database problem

- Phil Bernstein – 2003
  “Data exchange is the oldest database problem”

- **EXPRESS**: IBM San Jose Research Lab – 1977
  EXtraction, Processing, and REStructuring System
  for transforming data between hierarchical databases.

- Data Exchange underlies:
  - Data Warehousing, ETL (Extract-Transform-Load) tasks;
  - XML Publishing, XML Storage, …
Theoretical Aspects of Data Interoperability

Develop a conceptual framework for formulating and studying fundamental problems in data interoperability:

- Semantics of data integration & data exchange
- Algorithms for data exchange
- Complexity of query answering
Outline of the Talk

- Schema Mappings and Data Exchange
- Solutions in Data Exchange
  - Universal Solutions
  - The Core of the Universal Solutions
- Query Answering in Data Exchange
- Composing Schema Mappings
Schema Mappings

- Schema mappings:
  high-level, declarative assertions that specify the relationship between two schemas.

- Ideally, schema mappings should be
  - expressive enough to specify data interoperability tasks;
  - simple enough to be efficiently manipulated by tools.

- Schema mappings constitute the essential building blocks in formalizing data integration and data exchange.

- Schema mappings play a prominent role in Bernstein’s metadata management framework.
Schema Mappings & Data Exchange

- **Schema Mapping** \( M = (S, T, \Sigma) \)
  - **Source** schema \( S \), **Target** schema \( T \)
  - High-level, declarative assertions \( \Sigma \) that specify the relationship between \( S \) and \( T \).

- **Data Exchange** via the schema mapping \( M = (S, T, \Sigma) \)
  Transform a given **source** instance \( I \) to a **target** instance \( J \), so that \( <I, J> \) satisfy the specifications \( \Sigma \) of \( M \).
Solutions in Schema Mappings

**Definition**: Schema Mapping \( M = (S, T, \Sigma) \)

If \( I \) is a source instance, then a solution for \( I \) is a target instance \( J \) such that \( <I, J> \) satisfy \( \Sigma \).

**Fact**: In general, for a given source instance \( I \),

- No solution for \( I \) may exist
- or
- Multiple solutions for \( I \) may exist; in fact, infinitely many solutions for \( I \) may exist.
**Definition**: Schema Mapping \( M = (S, T, \Sigma) \)

- The **existence-of-solutions problem** \( \text{Sol}(M) \): (decision problem)
  Given a source instance \( I \), is there a solution \( J \) for \( I \)?

- The **data exchange problem associated with** \( M \): (function problem)
  Given a source instance \( I \), construct a solution \( J \) for \( I \), provided a solution exists.
Question: How are schema mappings specified?

Answer: Use logic. In particular, it is natural to try to use first-order logic as a specification language for schema mappings.

Fact: There is a fixed first-order sentence specifying a schema mapping $M^*$ such that $\text{Sol}(M^*)$ is undecidable.

Hence, we need to restrict ourselves to well-behaved fragments of first-order logic.
Embedded Implicational Dependencies

- **Dependency Theory**: extensive study of constraints in relational databases in the 1970s and 1980s.

- **Embedded Implicational Dependencies**: Fagin, Beeri-Vardi, …
  Class of constraints with a balance between high expressive power and good algorithmic properties:
  - **Tuple-generating dependencies** (tgds)
    Inclusion and multi-valued dependencies are a special case.
  - **Equality-generating dependencies** (egds)
    Functional dependencies are a special case.
Joint work with R. Fagin, R.J. Miller, and L. Popa

Studied data exchange between relational schemas for schema mappings specified by
- Source-to-target tgds
- Target tgds
- Target egds
The relationship between source and target is given by formulas of first-order logic, called

Source-to-Target Tuple Generating Dependencies (s-t tgds)

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

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

Example:

\[(\text{Student}(s) \land \text{Enrolls}(s,c)) \rightarrow \exists t \exists g (\text{Teaches}(t,c) \land \text{Grade}(s,c,g))\]
Schema Mapping Specification Language

- s-t tgds assert that:
  some SPJ source query is contained in some other SPJ target query

\[(\text{Student } (s) \land \text{Enrolls}(s,c)) \rightarrow \exists t \exists g (\text{Teaches}(t,c) \land \text{Grade}(s,c,g))\]

- s-t tgds generalize the main specifications used in data integration:
  - They generalize LAV (local-as-view) specifications:
    \[P(x) \rightarrow \exists y \psi(x, y), \text{ where } P \text{ is a source schema.}\]
  - They generalize GAV (global-as-view) specifications:
    \[\varphi(x) \rightarrow R(x), \text{ where } R \text{ is a target schema}\]
  - At present, most commercial II systems support GAV only.
In addition to source-to-target dependencies, we also consider target dependencies:

- **Target Tgds**: \( \varphi_T(x) \rightarrow \exists y \, \psi_T(x, y) \)

  Dept \((did, dname, mgr_id, mgr_name)\) \rightarrow Mgr \((mgr_id, did)\)

  (a target inclusion dependency constraint)

- **Target Equality Generating Dependencies (egds):**

  \( \varphi_T(x) \rightarrow (x_1 = x_2) \)

  \((\text{Mgr} \,(e, d_1) \land \text{Mgr} \,(e, d_2)) \rightarrow (d_1 = d_2)\)

  (a target key constraint)
Schema Mapping $M = (S, T, \Sigma_{st}, \Sigma_{t})$, where

- $\Sigma_{st}$ is a set of source-to-target tgds
- $\Sigma_{t}$ is a set of target tgds and target egds
Underspecification in Data Exchange

- **Fact:** Given a source instance, multiple solutions may exist.

- **Example:**
  Source relation $E(A,B)$, target relation $H(A,B)$
  \[ \Sigma: \quad E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y)) \]
  Source instance $I = \{E(a,b)\}$
  Solutions: **Infinitely** many solutions exist
  - $J_1 = \{H(a,b), H(b,b)\}$  \[ \text{constants:} \quad a, b, ... \]
  - $J_2 = \{H(a,a), H(a,b)\}$  \[ \text{variables (labelled nulls):} \quad X, Y, ... \]
  - $J_3 = \{H(a,X), H(X,b)\}$
  - $J_4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\}$
  - $J_5 = \{H(a,X), H(X,b), H(Y,Y)\}$
Main issues in data exchange

For a given source instance, there may be multiple target instances satisfying the specifications of the schema mapping. Thus,

- When more than one solution exist, which solutions are “better” than others?
- How do we compute a “best” solution?
- In other words, what is the “right” semantics of data exchange?
We introduced the notion of universal solutions as the “best” solutions in data exchange.

- By definition, a solution is universal if it has homomorphisms to all other solutions (thus, it is a “most general” solution).
- Constants: entries in source instances
- Variables (labeled nulls): other entries in target instances
- Homomorphism $h: J_1 \rightarrow J_2$ between target instances:
  - $h(c) = c$, for constant $c$
  - If $P(a_1,\ldots,a_m)$ is in $J_1$, then $P(h(a_1),\ldots,h(a_m))$ is in $J_2$
Universal Solutions in Data Exchange

Schema S

Schema T

Σ

I

J

Universal Solution

h₁

h₂

h₃

J₁

J₂

J₃

Homomorphisms

Solutions
Example - continued

Source relation $S(A,B)$, target relation $T(A,B)$

$\Sigma : E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y))$

Source instance $I = \{H(a,b)\}$

Solutions: Infinitely many solutions exist

- $J_1 = \{H(a,b), H(b,b)\}$ is not universal
- $J_2 = \{H(a,a), H(a,b)\}$ is not universal
- $J_3 = \{H(a,X), H(X,b)\}$ is universal
- $J_4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\}$ is universal
- $J_5 = \{H(a,X), H(X,b), H(Y,Y)\}$ is not universal
Universal solutions are analogous to most general unifiers in logic programming.

**Uniqueness up to homomorphic equivalence:**
If \( J \) and \( J' \) are universal for \( I \), then they are homomorphically equivalent.

**Representation of the entire space of solutions:**
Assume that \( J \) is universal for \( I \), and \( J' \) is universal for \( I' \). Then the following are equivalent:
1. \( I \) and \( I' \) have the same space of solutions.
2. \( J \) and \( J' \) are homomorphically equivalent.
Algorithmic Properties of Universal Solutions

**Theorem (FKMP):** Schema mapping $M = (S, T, \Sigma_{st}, \Sigma_t)$ such that:
- $\Sigma_{st}$ is a set of source-to-target tgds;
- $\Sigma_t$ is the union of a weakly acyclic set of target tgds with a set of target egds.

Then:
- Universal solutions exist if and only if solutions exist.
- $Sol(M)$, the existence-of-solutions problem for $M$, is in P.
- A *canonical* universal solution (if solutions exist) can be produced in polynomial time using the chase procedure.
Weakly Acyclic Sets of Tgds

Weakly acyclic sets of tgds contain as special cases:

- **Sets of full tgds**
  \[ \varphi_T(x) \rightarrow \psi_T(x), \]
  where \( \varphi_T(x) \) and \( \psi_T(x) \) are conjunctions of target atoms.

  **Example:** \( H(x,z) \land H(z,y) \rightarrow H(x,y) \land C(z) \)

  Full tgds express containment between relational joins.

- **Sets of acyclic inclusion dependencies**
  Large class of dependencies occurring in practice.
**The Smallest Universal Solution**

- **Fact:** Universal solutions need not be unique.
- **Question:** Is there a “best” universal solution?
- **Answer:** In joint work with R. Fagin and L. Popa, we took a “small is beautiful” approach:
  
  There is a smallest universal solution (if solutions exist); hence, the most compact one to materialize.

- **Definition:** The core of an instance $J$ is the smallest subinstance $J'$ that is homomorphically equivalent to $J$.

- **Fact:**
  - Every finite relational structure has a core.
  - The core is unique up to isomorphism.
The Core of a Structure

**Definition:** $J'$ is the core of $J$ if
- $J' \subseteq J$
- there is a hom. $h: J \to J'$
- there is **no** hom. $g: J \to J''$, where $J'' \subset J'$.
The Core of a Structure

**Definition:** $J'$ is the core of $J$ if
- $J' \subseteq J$
- there is a hom. $h: J \rightarrow J'$
- there is no hom. $g: J \rightarrow J''$, where $J'' \subset J'$.

**Example:** If a graph $G$ contains a $\triangle$, then $G$ is 3-colorable if and only if $\text{core}(G) = \triangle$.

**Fact:** Computing cores of graphs is an NP-hard problem.
Example - continued

Source relation E(A,B), target relation H(A,B)

\[ \Sigma : \ (E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y)) \]

Source instance \( I = \{E(a,b)\} \).

Solutions: Infinitely many universal solutions exist.

- \( J_3 = \{H(a,X), H(X,b)\} \) is the core.
- \( J_4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\} \) is universal, but not the core.
- \( J_5 = \{H(a,X), H(X,b), H(Y,Y)\} \) is not universal.
Core: The smallest universal solution

**Theorem (FKP):** \( M = (S, T, \Sigma_{st}, \Sigma_t) \) a schema mapping:

- All universal solutions have the same core.
- The core of the universal solutions is the smallest universal solution.
- If every target constraint is an egd, then the core is polynomial-time computable.

**Theorem (Gottlob – PODS 2005):** \( M = (S, T, \Sigma_{st}, \Sigma_t) \)

If every target constraint is an egd or a full tgd, then the core is polynomial-time computable.
Outline of the Talk

✓ Schema Mappings and Data Exchange

✓ Solutions in Data Exchange
  ✓ Universal Solutions
  ✓ The Core of the Universal Solutions

- Query Answering in Data Exchange

- Composing Schema Mappings
**Question:** What is the semantics of target query answering?

**Definition:** The certain answers of a query \( q \) over \( T \) on \( I \)

\[
certain(q,I) = \bigcap \{ q(J) : J \text{ is a solution for } I \}.
\]

**Note:** It is the standard semantics in data integration.
Certain Answers Semantics

\[
\text{certain}(q, I) = \bigcap \{ q(J) : J \text{ is a solution for } I \}.
\]
Computing the Certain Answers

**Theorem (FKMP):** Schema mapping $M = (S, T, \Sigma_{st}, \Sigma_t)$ such that:

- $\Sigma_{st}$ is a set of source-to-target tgds, and
- $\Sigma_t$ is the union of a weakly acyclic set of tgds with a set of egds.

Let $q$ be a union of conjunctive queries over $T$.

- If $I$ is a source instance and $J$ is a universal solution for $I$, then
  
  $\text{certain}(q, I) = \text{the set of all "null-free" tuples in } q(J)$.

  Hence, $\text{certain}(q, I)$ is computable in time polynomial in $|I|$: 
  1. Compute a canonical universal $J$ solution in polynomial time; 
  2. Evaluate $q(J)$ and remove tuples with nulls.

**Note:** This is a data complexity result ($M$ and $q$ are fixed).
Certain Answers via Universal Solutions

$q(J_1)$

$q(J_2)$

$q(J_3)$

$q(J)$

$\text{certain}(q,I) = \text{set of null-free tuples of } q(J)$. 

universal solution $J$ for $I$
Computing the Certain Answers

**Theorem (FKMP):** Schema mapping $M = (S, T, \Sigma_{st}, \Sigma_t)$ such that:
- $\Sigma_{st}$ is a set of source-to-target tgds, and
- $\Sigma_t$ is the union of a weakly acyclic set of tgds with a set of egds.

Let $q$ be a union of conjunctive queries with inequalities ($\neq$).
- If $q$ has at most one inequality per conjunct, then $\text{certain}(q, I)$ is computable in time polynomial in $|I|$ using a disjunctive chase.
- If $q$ is has at most two inequalities per conjunct, then $\text{certain}(q, I)$ can be coNP-complete, even if $\Sigma_t = \emptyset$. 
Universal Certain Answers

- Alternative semantics of query answering based on universal solutions.
- Certain Answers:
  “Possible Worlds” = Solutions
- Universal Certain Answers:
  “Possible Worlds” = Universal Solutions

**Definition:** Universal certain answers of a query q over T on I

\[
\text{u-certain}(q,I) = \cap \{ q(J) : J \text{ is a universal solution for } I \}.
\]

**Facts:**
- \(\text{certain}(q,I) \subseteq \text{u-certain}(q,I)\)
- \(\text{certain}(q,I) = \text{u-certain}(q,I)\), q a union of conjunctive queries
Computing the Universal Certain Answers

**Theorem (FKP):** Schema mapping $M = (S, T, \Sigma_{st}, \Sigma_t)$ such that:
- $\Sigma_{st}$ is a set of source-to-target tgds
- $\Sigma_t$ is a set of target egds and target tgds.

Let $q$ be an existential query over $T$.
- If $I$ is a source instance and $J$ is a universal solution for $I$, then

$$u\text{-certain}(q, I) = \text{the set of all “null-free” tuples in } q(\text{core}(J)).$$

- Hence, $u\text{-certain}(q, I)$ is computable in time polynomial in $|I|$ whenever the core of the universal solutions is polynomial-time computable.

**Note:** Unions of conjunctive queries with inequalities are a special case of existential queries.
Universal Certain Answers via the Core

$q(J_1)$

$q(J_2)$

$q(J_3)$

$q: existential$

$u\text{-}certain(q,I) = set\ of\ null-free\ tuples\ of\ q(\text{core}(J))$. 

universal solution $J$ for $I$
From Theory to Practice

- Clio/Criollo Project at IBM Almaden managed by Howard Ho.
  - Semi-automatic schema-mapping generation tool;
  - Data exchange system based on schema mappings.

- Universal solutions used as the semantics of data exchange.

- Universal solutions are generated via SQL queries extended with Skolem functions (implementation of chase procedure), provided there are no target constraints.

- Clio/Criollo technology is being exported to WebSphere II.
Some Features of Clio

- Supports nested structures
  - Nested Relational Model
  - Nested Constraints
- Automatic & semi-automatic discovery of attribute correspondence.
- Interactive derivation of schema mappings.
- Performs data exchange
Schema Mappings in Clio

Source Schema S

"conforms to"

data

Mapping Generation

Schema Mapping

Target Schema T

"conforms to"

Data exchange process (or SQL/XQuery/XSLT)
Outline of the Talk

✓ Schema Mappings and Data Exchange

✓ Solutions in Data Exchange
  ✓ Universal Solutions
  ✓ The Core of the Universal Solutions

✓ Query Answering in Data Exchange

- Composing Schema Mappings
  joint work with R. Fagin, L. Popa, and W.-C. Tan
Managing Schema Mappings

- Schema mappings can be quite complex.

- Methods and tools are needed to manage schema mappings automatically.

- **Metadata Management Framework** – Bernstein 2003 based on generic schema-mapping operators:
  - Composition operator
  - Inverse operator
  - Merge 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 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
  - However, their definition is in terms of maintaining the same certain answers relative to a class of queries.
  - 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:
  \[
  \text{Inst}(M) = \{ <I,J> \mid <I,J> \models \Sigma \}.
  \]

- **Definition: (FKPT)**
  A schema mapping $M_{13}$ is a composition of $M_{12}$ and $M_{23}$ if
  \[
  \text{Inst}(M_{13}) = \text{Inst}(M_{12}) \circ \text{Inst}(M_{23}),
  \]
  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 $<I_2,I_3> \models \Sigma_{23}$.

- **Note:** Also considered by S. Melnik in his Ph.D. thesis
**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.

**Definition:** The composition query of $M_{12}$ and $M_{23}$ is the set $\text{Inst}(M_{12}) \circ \text{Inst}(M_{23})$.
Issues in Composition of Schema Mappings

- The semantics of composition was the first main issue.

Some other key issues:

- Is the language of s-t tgds *closed under composition*?
  If $M_{12}$ and $M_{23}$ are specified by finite sets of s-t tgds, is $M_{12} \circ M_{23}$ also specified by a finite set of s-t tgds?

- If not, what is the “right” language for composing schema mappings?
## Composition: Expressibility & Complexity

<table>
<thead>
<tr>
<th>$M_{12}$</th>
<th>$M_{23}$</th>
<th>$M_{12} \circ M_{23}$</th>
<th>Composition Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma_{12}$</td>
<td>$\Sigma_{23}$</td>
<td>$\Sigma_{13}$</td>
<td>in PTIME</td>
</tr>
<tr>
<td>finite set of full s-t tgds</td>
<td>finite set of s-t tgds</td>
<td>finite set of s-t tgds</td>
<td>may not be definable: by any set of s-t tgds; in FO-logic; in Datalog</td>
</tr>
<tr>
<td>$\varphi(x) \rightarrow \psi(x)$</td>
<td>$\varphi(x) \rightarrow \exists y \psi(x, y)$</td>
<td>$\varphi(x) \rightarrow \exists y \psi(x, y)$</td>
<td>in NP; can be NP-complete</td>
</tr>
<tr>
<td>finite set of s-t tgds</td>
<td>finite set of (full) s-t tgds</td>
<td>may not be definable: by any set of s-t tgds; in FO-logic; in Datalog</td>
<td>in NP; can be NP-complete</td>
</tr>
<tr>
<td>$\varphi(x) \rightarrow \exists y \psi(x, y)$</td>
<td>$\varphi(x) \rightarrow \exists y \psi(x, y)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes:
- The expressions $\varphi(x)$ and $\psi(x, y)$ represent logical formulas in first-order logic.
- $\Sigma_1$ and $\Sigma_2$ are sets of formulae that are used to construct the composition query.
- $\circ$ denotes the composition of relations.
- PTIME stands for polynomial-time computable.
Employee Example

- $\Sigma_{12}$:
  - $\text{Emp}(e) \rightarrow \exists m \text{ Rep}(e,m)$

- $\Sigma_{23}$:
  - $\text{Rep}(e,m) \rightarrow \text{Mgr}(e,m)$
  - $\text{Rep}(e,e) \rightarrow \text{SelfMgr}(e)$

- **Theorem**: This composition is not definable by any finite set of s-t tgds.

- **Fact**: This composition is definable in a well-behaved fragment of second-order logic, called SO tgds, that extends s-t tgds with Skolem functions.
Employee Example - revisited

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

$\Sigma_{23}$:
- $\forall e \forall m \ ( \text{Rep}(e,m) \rightarrow \text{Mgr}(e,m) )$
- $\forall e \ ( \text{Rep}(e,e) \rightarrow \text{SelfMgr}(e) )$

**Fact:** The composition is definable by the SO-tgd

$\Sigma_{13}$:
- $\exists f \ ( \forall e \ ( \text{Emp}(e) \rightarrow \text{Mgr}(e,f(e)) ) \land$
  - $\forall e \ ( \text{Emp}(e) \land (e=f(e)) \rightarrow \text{SelfMgr}(e) ) )$
**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:**

$$\exists f ( (\forall e(\text{Emp}(e) \rightarrow \text{Mgr}(e,f(e))) \land \forall e(\text{Emp}(e) \land (e=f(e)) \rightarrow \text{SelfMgr}(e) ) )$$
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 schema mappings specified by SO-tgds, so that it produces universal solutions in polynomial time.
- For schema mappings specified by SO-tgds, the certain answers of target conjunctive queries are polynomial-time computable.
Synopsis of Schema Mapping Composition

- s-t tgds are **not** closed under composition.

- SO-tgds form a **well-behaved** fragment of second-order logic.
  - SO-tgds are closed under composition; they are a "**good**" language for composing schema mappings.
  - SO-tgds are "**chasable**": Polynomial-time data exchange with universal solutions.

- SO-tgds and the composition algorithm have been incorporated in Criollo’s **Mapping Specification Language (MSL)**.
Related Work and Extensions in this PODS

- G. Gottlob:  
  *Computing Cores for Data Exchange: Algorithms & Practical Solutions*

- A. Nash, Ph. Bernstein, S. Melnik:  
  *Composition of Mappings Given by Embedded Dependencies*

- A. Fuxman, Ph. Kolaitis, R.J. Miller, W.-C. Tan:  
  *Peer Data Exchange*

- M. Arenas & L. Libkin:  
  *XML Data Exchange: Consistency and Query Answering*
"Quelli che s'innamoran di pratica sanza scienza, son come 'l nocchiere ch'entra in navilio sanza timone o bussola, che mai ha certezza dove si vada"

Leonardo da Vinci, 1452-1519

"He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast."
Reduction from 3-Colorability

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

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

Let \( I_3 = \{ \texttt{(r,g)}, \texttt{(g,r)}, \texttt{(b,r)}, \texttt{(r,b)}, \texttt{(g,b)}, \texttt{(b,g)} \} \)

Given \( G=(V, E) \),
- let \( I_1 \) be the instance over \( S_1 \) consisting of the edge relation \( E \) of \( G \)

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

[Dawar98] showed that 3-colorability is not expressible in \( L_{\omega} \)
Algorithm Compose($M_{12}, M_{23}$)

- **Input**: Two schema mappings $M_{12}$ and $M_{23}$
- **Output**: A schema mapping $M_{13} = M_{12} \circ M_{23}$

Step 1: Split up tgds in $\Sigma_{12}$ and $\Sigma_{23}$
- $C_{12} = \text{Emp}(e) \rightarrow (\text{Mgr1}(e, f(e)))$
- $C_{23} =$
  - $\text{Mgr1}(e, m) \rightarrow \text{Mgr}(e, m)$
  - $\text{Mgr1}(e, e) \rightarrow \text{SelfMgr}(e)$

Step 2: Compose $C_{12}$ with $C_{23}$
- $\chi_1: \text{Emp}(e_0) \land (e=e_0) \land (m=f(e_0)) \rightarrow \text{Mgr1}(e, m)$
- $\chi_2: \text{Emp}(e_0) \land (e=e_0) \land (e=f(e_0)) \rightarrow \text{SelfMgr}(e)$

Step 3: Construct $M_{13}$
- Return $M_{13} = (S_1, S_3, \Sigma_{13})$ where
  - $\Sigma_{13} = \exists f(\exists e_0 \forall e \exists m \chi_1 \land \exists e_0 \exists e \chi_2)$