Reflections on Schema Mappings, Data Exchange, and Metadata Management

Phokion G. Kolaitis

UC Santa Cruz & IBM Research - Almaden
A Very Brief History

- Data exchange is the oldest database problem. Bernstein - 2003
- Yet, data exchange was not formalized until around 2000.
- Data exchange was formalized using schema mappings.
  - Schema Mappings as Query Discovery
    Miller, Haas, Hernández - VLDB 2000
  - Clio: A Semi-Automatic Tool for Schema Mapping
    Hernández, Miller, Haas - SIGMOD 2001
  - Data Exchange: Semantics and Query Answering
    Fagin, K . . ., Miller, Popa - ICDT 2003
- Extensive study of data exchange and schema mappings during the past 15 years.
Roadmap

Part I: Schema Mappings and Data Exchange
  ▸ Algorithmic and structural properties.

Part II: Operations on Schema Mappings
  ▸ Composing and inverting schema mappings.

Part III: Understanding and Deriving Schema Mappings
  ▸ Data examples to understand/derive schema mappings.
Transform data structured under a source schema $S$ into data structured a target schema $T$. 
Definition: A schema mapping is a triple $\mathcal{M} = (S, T, \mathcal{W})$ with

- $S$ is a source schema, $T$ is a target schema,
- $\mathcal{W}$ is a set of pairs $(I, J)$ with $I$ a source instance and $J$ a target instance.

Syntactically, a schema mapping is a triple $\mathcal{M} = (S, T, \Sigma)$ with

- $S$ is a source schema, $T$ is a target schema,
- $\Sigma$ is a set of constraints in some logical formalism expressing the relationship between $S$ and $T$.

$$(I, J) \models \Sigma \text{ if and only if } (I, J) \in \mathcal{W}.$$

Definition: A solution for a source instance $I$ with respect to $\mathcal{M}$ is a target instance $J$ such that $(I, J) \in \mathcal{W}$ (or, $(I, J) \models \Sigma$).
Algorithmic Problems for Schema Mappings

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathcal{W})$ be a fixed schema mapping.

Existence of Solutions Problem:

Given a source instance $I$,

- determine whether or not a solution for $I$ w.r.t. $\mathcal{M}$ exists;
- if so, compute a "good" solution for $I$ w.r.t. $\mathcal{M}$.

Certain Answers Problem:

Let $q$ be a fixed query over the target schema $\mathbf{T}$.

Given a source instance $I$,

- compute the certain answers of $q$ on $I$ w.r.t. $\mathcal{M}$

$$
\text{cert}(q, I, \mathcal{M}) = \bigcap \{ q(J) : J \text{ is a solution for } I \text{ w.r.t. } \mathcal{M} \}.
$$
Question: What is a "good" schema-mapping language?

Ideally, a "good" schema-mapping language should have

- sufficient expressive power to express interesting data-transformation tasks;
- tractable algorithmic behavior.

Fact: The existence-of-solutions problem is undecidable for schema mappings specified by first-order sentences.

- Reduction from the finite validity problem for FO.
Basic Tasks for a Schema-Mapping Language

- **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.,**
  \[ \forall x, y, z(E(x, z) \land F(z, y) \rightarrow \exists w T(x, y, z, w)) \]
Global-and-Local-As-View Constraints

Definition: A Global-and-Local-as-View (GLAV) constraint is a first-order sentence of the form

$$\forall x (\varphi(x) \rightarrow \exists y \psi(x, y)),$$

where

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

Example:

$$\forall c, i, s \,(\text{ENROLLS}(s, c) \land \text{TEACHES}(i, c) \rightarrow \exists g \, \text{GRADES}(s, c, g))$$

Fact:
Each basic task (projection, decomposition, join, ...) can be expressed by a GLAV constraint.
GLAV, GAV, LAV Constraints and Mappings

- Global-and-Local-as-View (GLAV) constraint
  \[ \forall x (\varphi(x) \rightarrow \exists y \psi(x, y)) \]

- Global-As-View (GAV) constraint
  \[ \forall x (\varphi(x) \rightarrow R(x)), \text{ where } R \text{ is a target relation.} \]
  - Copy, Projection, Join

- Local-As-View (LAV) constraint
  \[ \forall x (P(x) \rightarrow \exists y \psi(x, y)), \text{ where } P \text{ is a source relation.} \]
  - Copy, Column Augmentation, Decomposition

- A GLAV mapping is a schema mapping \( \mathcal{M} = (S, T, \Sigma) \), where \( \Sigma \) is a finite set of GLAV constraints.

- Similarly, for the notions of a GAV mapping and a LAV mapping.
Structural Properties of GLAV Mappings

Theorem (Fagin, K . . ., Miller, Popa - 2003)

Let $\mathcal{M} = (\mathcal{S}, \mathcal{T}, \Sigma)$ be a GLAV mapping.

- $\mathcal{M}$ admits universal solutions
  For every source instance $I$, there is a solution $J^*$ for $I$ such that for every solution $J$ for $I$, there is a homomorphism from $J^*$ to $J$ that is the identity on elements from $I$.
    - Moreover, such a $J^*$ can be computed in polynomial time in the size of $I$ via the chase procedure.

- $\mathcal{M}$ allows for CQ-rewriting
  For every conjunctive query $q$ over $\mathcal{T}$, there is a union $q'$ of conjunctive queries over $\mathcal{S}$ such that
  $$\text{cert}(q, I, \mathcal{M}) = q'(I).$$
    - In particular, the certain answers of $q$ can be computed in polynomial time in the size of $I$. 
Universal Solutions in Data Exchange

Diagram:
- Schema $S$
- Schema $T$
- Universal Solution
- Homomorphisms $h_1$, $h_2$, $h_3$
- Solutions $J_1$, $J_2$, $J_3$
Proposition:

- Every GLAV mapping $M$ is closed under target homomorphisms
  If $J$ is a solution for $I$ and there is a homomorphism from $J$ to $J'$ that is the identity on elements of $I$, then $J'$ is a solution for $I$.

- Every GAV mapping $M$ is closed under target intersections
  If $J$ and $J'$ are solutions for $I$, then $J \cap J'$ is a solution for $I$.

- Every LAV mapping $M$ is closed under unions
  If $J$ is a solution for $I$ and if $J'$ is a solution for $I'$, then $J \cup J'$ is a solution for $I \cup I'$. 
Characterizing GAV and LAV Mappings

Theorem (ten Cate and K . . . - 2009)
Let $\mathcal{M} = (\mathcal{S}, \mathcal{T}, \mathcal{W})$ be a schema mapping. Then the following statements are equivalent:

1. $\mathcal{M}$ is logically equivalent to a GAV mapping (respectively, $\mathcal{M}$ is logically to a LAV mapping).

2. $\mathcal{M}$ has the following properties:
   - $\mathcal{M}$ admits universal solutions;
   - $\mathcal{M}$ allows for CQ-rewriting;
   - $\mathcal{M}$ is closed under target homomorphisms;
   - $\mathcal{M}$ is closed under target intersections (respectively, $\mathcal{M}$ is closed under unions).
Theorem (ten Cate and K . . . - 2009)
Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathcal{W})$ be a schema mapping. Then the following statements are equivalent:

1. $\mathcal{M}$ is logically equivalent to a GLAV mapping.
2. $\mathcal{M}$ has the following properties:
   - $\mathcal{M}$ admits universal solutions;
   - $\mathcal{M}$ allows for CQ-rewriting;
   - $\mathcal{M}$ is closed under target homomorphisms;
   - $\mathcal{M}$ is $n$-modular, for some $n \geq 1$.

Definition
A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathcal{W})$ is $n$-modular if whenever $(I, J) \notin \mathcal{W}$, there is some $I' \subseteq I$ such that $|I'| \leq n$ and $(I', J) \notin \mathcal{W}$. 
## Structural Characterizations: Summary

<table>
<thead>
<tr>
<th>Type of Schema-Mapping</th>
<th>Characterizing Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAV Mapping</td>
<td>admits universal solutions \ allows for CQ-rewriting \ closed under target homomorphisms \ closed under target intersections</td>
</tr>
<tr>
<td>LAV Mapping</td>
<td>admits universal solutions \ allows for CQ-rewriting \ closed under target homomorphisms \ closed under unions</td>
</tr>
<tr>
<td>GLAV Mapping</td>
<td>admits universal solutions \ allows for CQ-rewriting \ closed under target homomorphisms \ ( n )-modular, for some ( n \geq 1 )</td>
</tr>
</tbody>
</table>
Managing Schema Mappings via Operators

- 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.

- Extensive study of the Composition operator and the Inverse operator.
Problem:

- 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 $M_{13} = M_{12} \circ M_{23}$.

- But, what does it mean to say that $M_{13}$ is "equivalent" to the composition of $M_{12}$ and $M_{23}$?
Semantics of Composition

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

- **Composing Mappings among Data Sources**  Madhavan and Halevy - 2003
  - First to propose a semantics for composition
  - Notion of composition relative to a class of queries
  - **CQ-composition**: relative to the class of conjunctive queries

- **Set-theoretic semantics of composition**  
  Fagin, K . . ., Popa, Tan - 2004, Melnik - 2004
Set-theoretic Semantics of Composition

Recall that
- a syntactically specified $\mathcal{M} = (S, T, \Sigma)$
  is identified with
- the semantically specified $\mathcal{M} = (S, T, \mathcal{W}(\mathcal{M}))$, where
  $\mathcal{W}(\mathcal{M}) = \{(I, J) : (I, J) \models \Sigma\}$.

Definition (FKPT - 2004, Melnik - 2004)
A schema mapping $\mathcal{M}_{13} = (S_1, S_3, \Sigma_{13})$ is the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ of $\mathcal{M}_{12} = (S_1, S_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (S_2, S_3, \Sigma_{23})$ if

$\mathcal{W}(\mathcal{M}_{13}) = \mathcal{W}(\mathcal{M}_{12}) \circ \mathcal{W}(\mathcal{M}_{23})$,

i.e.,
- $(I_1, I_3) \in \mathcal{W}(\mathcal{M}_{13})$
  if and only if
- there is some $I_2$ such that $(I_1, I_2) \in \mathcal{W}(\mathcal{M}_{12})$ and $(I_2, I_3) \in \mathcal{W}(\mathcal{M}_{23})$. 
The Language of Composition

Questions:

- Is the language of GLAV constraints closed under composition?
  In other words:

- If $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ are GLAV mappings, is $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ a GLAV mapping as well?

- If not, what is the *right* language for composing schema mappings?
The Language of Composition

Theorem (Fagin, K ..., Popa, Tan - 2004)

- GAV mappings are closed under composition.
- GLAV mappings are not closed under composition.
- In fact, there are GLAV mappings $M_{12}$ and $M_{23}$ whose composition $M_{12} \circ M_{23}$ is not expressible even in least fixed-point logic LFP.

Question:
What is the right language for composing GLAV mappings?
Towards the “Right” Language for Composition

Motivating Example:

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

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

Theorem:

- The composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is not definable by any set (finite or infinite) of GLAV constraints.

- The composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is definable by the following Second-Order GLAV constraint:

  $\exists f (\forall e (\text{Emp}(e) \rightarrow \text{Mgr}(e, f(e))) \land$
  $\forall e (\text{Emp}(e) \land (e = f(e) \rightarrow \text{SelfMgr}(e))))$
Second Order GLAV Constraints

**Definition:** Let $S$ be a source schema and $T$ a target schema. A **Second-Order GLAV constraint** (SO GLAV) is a formula of the form:

$$\exists f_1 \cdots \exists f_n ((\forall x_1 (\varphi_1(x_1) \rightarrow \psi_1(x_1))) \land \cdots \land (\forall x_n (\varphi_n(x_1) \rightarrow \psi_n(x_n)))),$$

where

- Each $f_i$ is a function symbol.
- Each $\varphi_i$ is a conjunction of atoms from $S$ and **equalities** of terms.
- Each $\psi_i$ is a conjunction of atoms from $T$.

**Example:** $\exists f (\forall e (\text{Emp}(e) \rightarrow \text{Mgr}(e, f(e))) \land \forall e (\text{Emp}(e) \land (e = f(e) \rightarrow \text{SelfMgr}(e))))$
Data Exchange via Second-Order GLAV Constraints

Theorem (Fagin, K ..., Popa, Tan - 2004)

- SO GLAV mappings are closed under composition.
- The chase procedure can be extended to SO GLAV mappings; in particular, it produces universal solutions in polynomial time.
- Every SO GLAV mapping is the composition of finitely many GLAV mappings (in fact, just two).

Conclusion: SO GLAV constraints are the right language for the composition of GLAV mappings.

Note: SO GLAV constraints and the composition algorithm are used in the IBM InfoSphere Information Server.
Definition: A plain SO GLAV constraint is a SO GLAV constraint with no equalities and no nested function terms.

Theorem (Arenas, Pérez, Reutter, Riveros - 2013)
Plain SO GLAV constraints are the right language for the CQ-composition of GLAV mappings.

Open Problem 1:
- Is there a structural characterization of SO GLAV mappings?
- Is there a structural characterization of plain SO GLAV mappings?
Inverting Schema Mappings

Problem: Given a schema mapping $M$, find a schema mapping $M^*$ that “undoes” what $M$ did.
Exact Inverses of Schema Mappings

Definition: Fagin - 2006
\( M^* \) is an inverse of \( M \) if \( M \circ M^* = Id \), where \( Id \) is the identity schema mapping specified by copy constraints.

Note: Schema mappings may entail inherent information loss.

- Union Schema Mapping

\[
\forall x (P(x) \rightarrow Q(x)) \\
\forall x (R(x) \rightarrow Q(x))
\]

Fact: Inverses of GLAV mappings rarely exist.
Approximate Inverses of Schema Mappings

Several different approaches, including:

- **Quasi-inverse**
  Fagin, K . . ., Popa, Tan - 2007

- **Maximum Recovery**
  Arenas, Pérez, Riveros - 2008

- **Chase Inverse**
  Fagin, K . . ., Popa, Tan - 2011

**Note:**

- **Maximum recoveries** have better properties than other notions of approximate inverses do.

- In particular, every plain SO GLAV mapping has a maximum recovery (hence, so does every GLAV mapping).
Combining Composition and Inversion

Fact:

- The language for expressing maximum recoveries involves disjunctive constraints.
- No definitive notion of an inverse has emerged.

Open Problem 2:

- Find a useful notion of inverse and a language for expressing it, so that the language is closed under compositions and inversions of GLAV mappings.
Fact: Schema evolution can be analyzed using the composition operator and the inverse operator.
Schema Mappings Can Be Complex

Map 2:
for sm2x0 in S0.dummy_COUNTRY_4
  exists tm2x0 in S27.dummy_country_10, tm2x1 in S27.dummy_organiza_13
  where tm2x0.country_member_id=tm2x1.organization_id,
  satsf sm2x0.COUNTRY.AREA=tm2x0.country.area, sm2x0.COUNTRY.CAPITAL=tm2x0.country.capital,
  sm2x0.COUNTRY.CODE=tm2x0.country.id, sm2x0.COUNTRY.NAME=tm2x0.country.name,
  sm2x0.COUNTRY.POPULATION=tm2x0.country.population,

Map 3:
for sm3x0 in S0.dummy_GEO_RIVE_23, sm3x1 in S0.dummy_RIVER_24,
  sm3x2 in S0.dummy_PROVINCE_5
  where sm3x0.GEO_RIVER.RIVER=sm3x1.RIVER.NAME, sm3x2.PROVINCE.NAME=sm3x0.GEO_RIVER.PROVINCE,
  sm3x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
  exists tm3x0 in S27.dummy_river_24, tm3x1 in tm3x0.river.dummy_located_23,
  tm3x4 in S27.dummy_country_10, tm3x5 in tm3x4.country.dummy_province_9,
  tm3x6 in S27.dummy_organiza_13
  where tm3x4.country_member=tm3x6.organization.id, tm3x5.province.id=tm3x1.located.province,
  tm2x0.country.id=tm3x1.located.country,
  satsf sm2x0.COUNTRY.AREA=tm3x4.country.area, sm2x0.COUNTRY.CAPITAL=tm3x4.country.capital,
  sm2x0.COUNTRY.CODE=tm3x4.country.id, sm2x0.COUNTRY.NAME=tm3x4.country.name,
  sm2x0.COUNTRY.POPULATION=tm3x4.country.population, sm3x1.RIVER.LENGTH=tm3x0.river.length,
  sm3x0.GEO_RIVER.RIVER.COUNTRY=tm3x1.located.country, sm3x0.GEO_RIVER.PROVINCE=tm3x1.located.province,
  sm3x1.RIVER.NAME=tm3x0.river.name ),

Map 4:
for sm4x0 in S0.dummy_GEO_ISLA_25, sm4x1 in S0.dummy_ISLAND_26,
  sm4x2 in S0.dummy_PROVINCE_5
  where sm4x0.GEO_ISLAND.ISLAND=sm4x1.ISLAND.NAME, sm4x2.PROVINCE.NAME=sm4x0.GEO_ISLAND.PROVINCE,
  sm4x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
  exists tm4x0 in S27.dummy_island_25, tm4x1 in tm4x0.island.dummy_located_25,
  tm4x4 in S27.dummy_country_10, tm4x5 in tm4x4.country.dummy_province_9,
  tm4x6 in S27.dummy_organiza_13
  where tm4x4.country_member=tm4x6.organization.id, tm4x5.province.id=tm4x1.located.province,
  tm2x0.country.id=tm4x1.located.country,
  satsf sm2x0.COUNTRY.AREA=tm4x4.country.area, sm2x0.COUNTRY.CAPITAL=tm4x4.country.capital,
  sm2x0.COUNTRY.CODE=tm4x4.country.id, sm2x0.COUNTRY.NAME=tm4x4.country.name,
  sm2x0.COUNTRY.POPULATION=tm4x4.country.population, sm4x1.ISLAND.AREA=tm4x0.island.area,
  sm4x1.ISLAND.COORDINATESLAT=tm4x0.island.latitude, sm4x0.GEO_ISLAND.COUNTRY=tm4x1.located.country,
  sm4x0.GEO_ISLAND.PROVINCE=tm4x1.located.province, sm4x1.ISLAND.COORDINATESLONG=tm4x0.island.longitude,
  sm4x1.ISLAND.NAME=tm4x0.island.name ),

Map 5:
for sm5x0 in S0.dummy_GEO_SEA_19, sm5x1 in S0.dummy_SEA_20,
  sm5x2 in S0.dummy_PROVINCE_5
  where sm5x2.PROVINCE.NAME=sm5x0.GEO_SEA.PROVINCE, sm5x0.GEO_SEA.SEA=sm5x1.SEA.NAME,
  sm5x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
  exists tm5x0 in S27.dummy_sea_19, tm5x1 in tm5x0.sea.dummy_located_18,
  tm5x4 in S27.dummy_country_10, tm5x5 in tm5x4.country.dummy_province_9,
  tm5x6 in S27.dummy_organiza_13
  where tm5x4.country_member=tm5x6.organization.id, tm5x5.province.id=tm5x1.located.province,
  tm2x0.country.id=tm5x1.located.country,
  satsf sm2x0.COUNTRY.AREA=tm5x4.country.area, sm2x0.COUNTRY.CAPITAL=tm5x4.country.capital,
Understanding and Deriving Schema Mappings

Idea:
Use data examples to understand/derive schema mappings

Theme I: From Syntax to Semantics
Understand schema mappings using data examples.

Theme II: From Semantics to Syntax:
Derive schema mappings using data examples.
Data Examples and Universal Examples

Definition: Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping.

- A data example for $\mathcal{M}$ is a pair $(I, J)$ such that $J$ is a solution for $I$ w.r.t. $\mathcal{M}$ (i.e., $(I, J) \models \Sigma$).
- A universal example for $\mathcal{M}$ is a pair $(I, J)$ such that $J$ is a universal solution for $I$ w.r.t. $\mathcal{M}$.

Note: The space of data examples and the space of universal examples are typically infinite.

Question: Can a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be captured by finitely many data examples or by finitely many universal examples?
From Syntax to Semantics: Unique Characterizations

Definition: Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping, $\mathbf{U}$ set of universal examples for $\mathcal{M}$, and $\mathbf{C}$ a class of GLAV constraints.

- $\mathbf{U}$ uniquely characterizes $\mathcal{M}$ w.r.t. $\mathbf{C}$ if for every schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ such that $\Sigma' \subseteq \mathbf{C}$ and $\mathbf{U}$ is a set of universal examples for $\mathcal{M}'$, we have that $\Sigma \equiv \Sigma'$.

Note:
- Unique characterizability via finitely many positive/negative examples implies unique characterizability via finitely many universal examples, but not vice-versa.
From Syntax to Semantics: Unique Characterizations

Theorem (Alexe, ten Cate, K . . . , Tan - 2011)

- Every LAV mapping is uniquely characterizable by a finite set of universal examples w.r.t. to LAV constraints.
- Criterion for a GAV mapping to be uniquely characterizable by a finite set of universal examples w.r.t. GAV constraints.
- The associated decision problem for GAV mappings is NP-complete.

Open Problem 3:

- Find criteria for a GLAV mapping to be uniquely characterizable by a finite set of universal examples w.r.t. GLAV constraints.
- What is the exact complexity of the associated decision problem for GLAV mappings?
From Semantics to Syntax: Derivations

Interactive Derivation and Refinement of Schema Mappings

- Fitting Algorithm - EIRENE System
  Alexe, ten Cate, K . . ., Tan - 2011

- Interacting Mapping Specification with Exemplar Tuples - IMS System
  Bonifati, Comignani, Coquery, Thion - 2017

Deriving Optimal Schema Mappings from Data Examples

- Cost Model for Optimal Repairs of Schema Mappings
  Gottlob and Senellart - 2010

- Cost Model for Mapping Selection via Data Examples
  Kimmig, Memory, Miller, Getoor - 2017
From Semantics to Syntax: Learning

Theorem (ten Cate, Dalmau, K . . . - 2012)
GAV mappings are efficiently learnable in Angluin’s model with membership and equivalence queries.

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Active Learning of GAV Mappings

Open Problem 4:

- Learnability of LAV mappings.
- Learnability of GLAV mappings.
Topics Not Covered - Partial List

- Richer schema mappings:
  - Schema mappings with target constraints
  - Schema mappings with arithmetic constraints
  - Schema mappings with bi-directional constraints.

- Alternative notions of solutions - Libkin 2006
- Alternative notions of certain answers
- Beyond conjunctive queries (non-monotonic, aggregate)
- XML Data Exchange - Arenas and Libkin 2005
- Benchmarks
  - STBenchmark - Alexe, Tan, Velegrakis 2008
  - iBench - Arocena, Glavic, Ciucanu, Miller 2015
Synopsis

- Mature body of research during the past 15 years.
- A case study of logic in computer science, but also of logic from computer science.
- Theory and practice have informed each other.

Challenges and Outlook

- Several key technical problems remain open.
- Little penetration of advanced technical findings to practice.
- Benchmarks and open-source systems for data exchange need to be developed.