Structural characterizations of schema mapping languages

Balder ten Cate
INRIA and ENS Cachan

(research done while visiting IBM Almaden and UC Santa Cruz)

Joint work with **Phokion Kolaitis** (ICDT’09)

Schema mappings

Schema mapping: set of declarative assertions specifying the relationship between two DB schemas.

Example:

- **Source schema S**: { Emp(*Name*, *Dept*, *SSN*), ... }
- **Target schema T**: { Staff(*Name*, *Dept*, *Phone*), ... }
- **Specifications Σ of the schema mapping**:

$$\{ \ \forall xyz \ (\ \mathrm{Emp}(x,y,z) \rightarrow \ \exists u.\mathrm{Staff}(x,y,u) \ ) \ \}$$

Data exchange and data integration

Schema mappings are used in data interoperability tasks such as data exchange and data integration.

Data exchange:

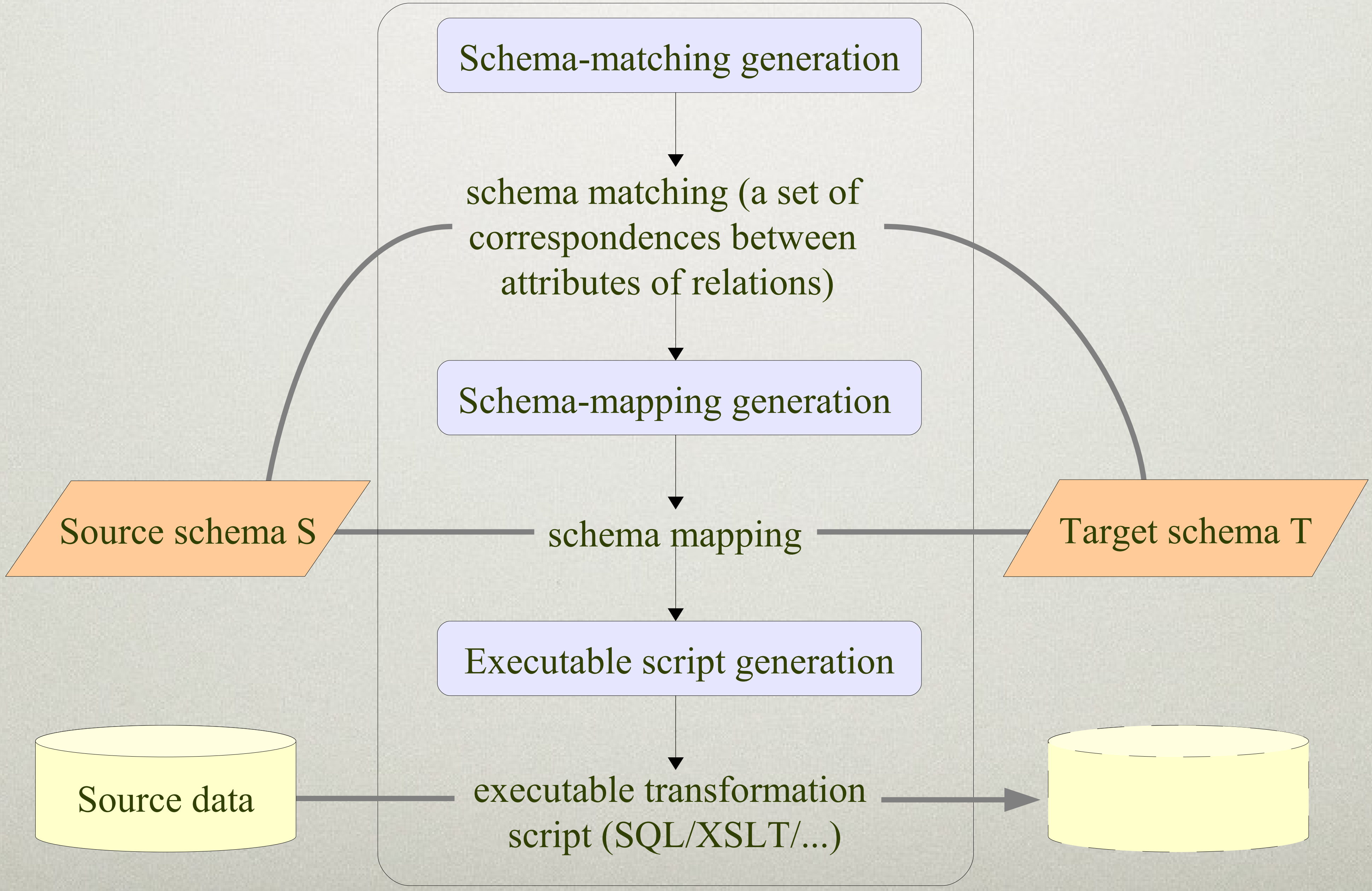
- Given a schema mapping and a source instance, compute a solution: a target instance satisfying the constraints.

Data integration:

- Given a schema mapping, a source instance I, and a target query q, compute the certain answers $\bigcap\{q(J) | J \text{ a solution of } I\}$.

Architecture of Clio

- Schema mappings provide a middle-layer in-between schema matchings and transformation scripts, having a well-defined semantics.



Schema mapping languages

Many schema mappings languages have been proposed:

- s-t tgds
- full s-t tgds
- LAV s-t tgds
- SO tgds
- nested s-t tgds

These are fragments of FO logic or of SO logic.

Why so many formalisms?

Each formalism has its own virtues. For example,

- Schema mappings specified by s-t tgds
    - admit universal solutions (each source instance has a universal solution: a solution with a homomorphism into every other solution)
    - allow for CQ rewriting (each target CQ can be rewritten to a source UCQ computing the certain answers)
- Schema mappings specified using only full s-t tgds, or only LAV s-t tgds, satisfy further desirable properties.

Characterizing schema mapping languages

Idea of the paper: to turn the tables and characterize each schema mapping language in terms of such properties.

"*a schema mapping is definable by L-constraints iff it satisfies X*"

$L \in$ { s-t tgds, full s-t tgds, LAV s-t tgds, ... },

$X \subseteq$ {admit-universal-solutions, allow-for-CQ-rewriting, ... }

Motivations

Understanding the expressive power of schema mapping languages.

Understanding the scope of basic techniques in data exchange and data integration

A check-list for definability (to test if a mapping is definable in a language, check that it satisfies the list of properties).

- E.g., can effectively test whether a schema mapping specified by s-t tgds can be defined using only LAV s-t tgds, etc.

Outline

Outline of remainder of talk:

1. Characterizations of schema mapping languages
2. Complexity of testing definability

Schema mappings: Abstract definition

Definition: a schema mapping is a triple ($\mathbf{S}$,$\mathbf{T}$,W), with W is a class of pairs (I,J) of instances for $\mathbf{S}$ and $\mathbf{T}$, respectively, invariant for isomorphisms (if $(I,J) \in W$ and $(I,J) \cong (I',J')$, then $(I',J') \in W$).

If $(I,J) \in W$, then J is called a solution for I.

Definition: a FO schema mapping is a schema mapping defined by a finite FO theory over $\mathbf{S} \cup \mathbf{T}$.

Schema mapping languages

Source-to-target tuple generating dependency (s-t tgd): a FO sentence $\forall \mathbf{x}.(\varphi_S \rightarrow \exists \mathbf{y}.\psi_T)$, where

- φ_S a conjunction of atomic formulas over **S**
- ψ_T a conjunction of atomic formulas over **T**
- every x_i occurs in φ_S.

Full s-t tgd: an s-t tgd without $\exists$-quantifiers

LAV s-t tgd: an s-t tgd in which φ_S is an atomic formula.

These roughly correspond to GLAV[CQ], GAV[CQ], and LAV[CQ] constraints under the sound semantics.

Properties (i)

Every schema mapping specified by a finite set of s-t tgds ...

- **is closed under target homomorphisms**: if J is a solution for I and h : J → J’ is a homomorphism constant on dom(I), then J’ is also a solution for I.
- **admits universal solutions**: every source instance I has a universal solution, i.e., a solution that has a homomorphism into every other solution (constant on dom(I)).
- **allows for CQ rewriting**: for every conjunctive query q there is a union of conjunctive queries q’ that computes the certain answers of q.

Properties (ii)

1+2 provide the foundation for data exchange. They imply:

- the (infinite) space of all solutions of a source instance is captured by a single solution, namely any universal solution.
- Every source instance has a core universal solution.

3 provides the foundation for data integration. It implies:

- certain answers can be computed in LogSpace (data complexity).

Properties (iii)

- Every schema mapping specified by a finite set of LAV s-t tgds...
  4. **is closed under union**: if J_1 is a solution for I_1, and J_2 for I_2, then $J_1 \cup J_2$ it is a solution for $I_1 \cup I_2$.
- Every schema mapping specified by a finite set of full s-t tgds...
  5. **is closed under target intersection**: if J_1 and J_2 are solutions for I, then $J_1 \cap J_2$ is.

LAV s-t tgds

Theorem 1: A schema mapping is definable by a finite set of LAV s-t tgds iff it

1. is closed under target homomorphisms
2. admits universal solution
3. allows for CQ rewriting
4. is closed under union

Full s-t tgds

Theorem 2: A schema mapping is definable by a finite set of full s-t tgds iff it

1. is closed under target homomorphisms
2. admits universal solution
3. allows for CQ rewriting
5. is closed under target intersection.

arbitrary s-t tgds (i)

Recall: every schema mapping definable by a finite set of s-t tgds

1. is closed under target homomorphisms
2. admits universal solution
3. allows for CQ rewriting

Theorem. Every schema mapping satisfying 1,2,3 is defined by a possibly infinite set of s-t tgds.

This is not a characterization! Some FO schema mappings satisfying 1,2,3 are not definable by a finite set of s-t tgds. Example: $\forall x. \exists y. \forall z. (Rxz \rightarrow Syz)$

Arbitrary s-t tgds (ii)

Extra condition needed:

6. ***n*-Modularity**: if J is a solution for each I’ $\subseteq$ I with |I’|≤ *n* then J is a solution for I.

Theorem 3: A schema mapping is definable by a finite set of s-t tgds iff it satisfies

- closure under target homomorphisms
- admitting universal solution
- reflecting source homomorphisms
- *n*-modularity for some *n* > 0

Summary

Schema mapping language	Characterizing properties
LAV s-t tgds	1. Closure under target homomorphisms 2. Admitting universal solutions 3. Allowing for CQ rewriting *4. Closure under union*
Full s-t tgds	1. Closure under target homomorphisms 2. Admitting universal solutions 3. Allowing for CQ rewriting *5. Closed under target intersection*
s-t tgds	1. Closure under target homomorphisms 2. Admitting universal solutions 3. Allowing for CQ rewriting *6. n-modularity (for some n>0)*

Some further variations

For FO schema mappings,

- the property of allowing for CQ rewriting can be replaced by the property of reflecting source homomorphisms:

  If *h: I-> I'* and *J'* is a solution for *I'* then *h* extends to a homomorphism from any universal solution of *I* to *J'*

which has a more "geometric" feeling to it and can be easier to check.

- *n*-modularity can be replaced by having bounded block size.

About the proofs

- Classical model theory provides techniques for characterizing the expressive power of logical languages
- The challenge is to obtain characterizations in the finite (Finite Model Theory), which requires different techniques.
- An important recent result in Finite Model Theory: Rossman's homomorphism preservation theorem (UCQs are precisely the FO queries preserved under homomorphisms).
- Some of our proofs use a lemma of Rossman, others use more elementary arguments.

Open question: characterize other languages

- SO tgds,
- nested s-t tgds (next slide)
- target dependencies (cf. Makowsky-Vardi ’86)
- ...

Nested s-t tgds

A very natural extension of s-t tgds used in the IBM Almaden data exchange system Clio:

Nested s-t tgds: FO sentences $\forall \mathbf{x}.(\varphi_S \rightarrow \exists \mathbf{y}.\psi_T)$, where

- φ_S a conjunction of atomic formulas over **S**
- ψ_T a conjunction of atomic formulas over **T** **and/or nested s-t tgds (possibly having x,y as FVs).**
- every x_i occurs in φ_S.

Fact: all schema mappings defined by a finite set of nested s-t tgds satisfy 1,2,3.

Question: are nested s-t tgds characterized by 1,2,3?

Part 2: Complexity of expressibility

Given a schema mapping specified by s-t tgds, can it be defined using only LAV s-t tgds?

- By our characterization, equivalent to closure under union.
- Can be effectively tested.

Similarly for full s-t tgds.

Input schema mapping language	Desired schema mapping language	Complexity of testing definability
s-t tgds	full s-t tgds	**NP-complete**
s-t tgds	LAV s-t tgds	**NP-complete**
full s-t tgds	LAV s-t tgds	**PTIME**
LAV s-t tgds	full s-t tgds	**NP-complete**

In each case, an equivalent schema mapping can be constructed in PTIME if it is known to exist.

Summary

1. we characterized
   - s-t tgds
   - LAV s-t tgds
   - full s-t tgds

in terms of natural properties, e.g., admitting universal solutions.

1. we used our characterizations to derive complexity results for testing definability in restricted fragments.
2. in the paper, we also consider BP-style (instance level) definability.

Future directions

- Our investigations point to nested s-t tgds as a natural schema mapping language (received little attention in the literature so far)

- What are suitable schema mapping languages for XML
  - There are proposals, but no general agreement yet
  - Structural characterizations would help

- Schema mapping optimization is arising as a new topic:
  - Rewrite schema mappings into 'equivalent' ones that are more efficient or otherwise more well-behaved
  - Different notions of equivalent: logical equivalence, data-exchange equivalence, conjunctive-query equivalence.

Thank You!