Schema Mappings and Data Examples

Balder ten Cate
UC Santa Cruz
and LogicBlox

Phokion Kolaitis
UC Santa Cruz
and IBM Research - Almaden

Wang-Chiew Tan
UC Santa Cruz

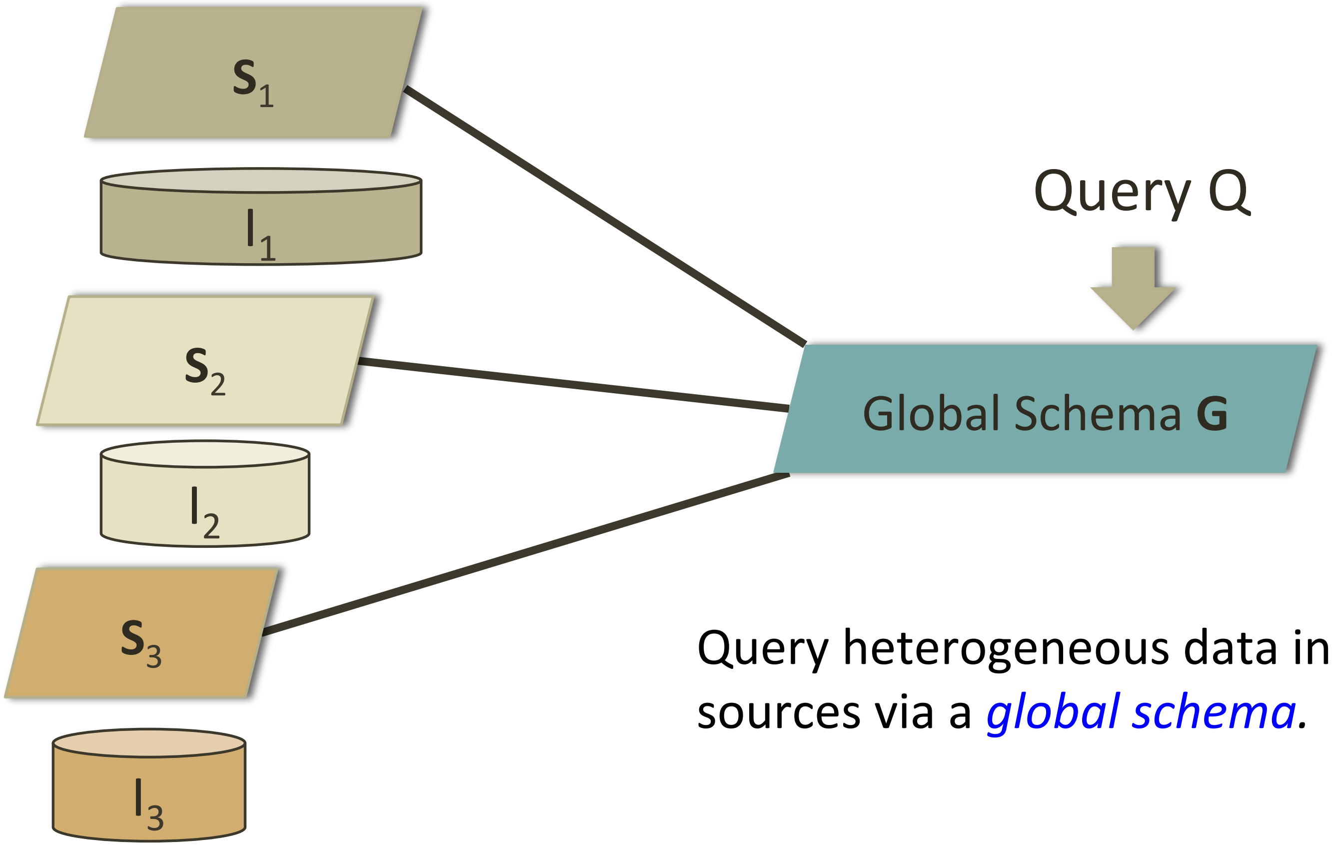
The Data Interoperability Challenge

- Data may be
  - *distributed* at several different locations.
  - *heterogeneous* in representation (relational, JSON, …).

- How can we uniformly access and manipulate data from these data sources?

- Two main approaches:
  1. Data integration
  2. Data exchange

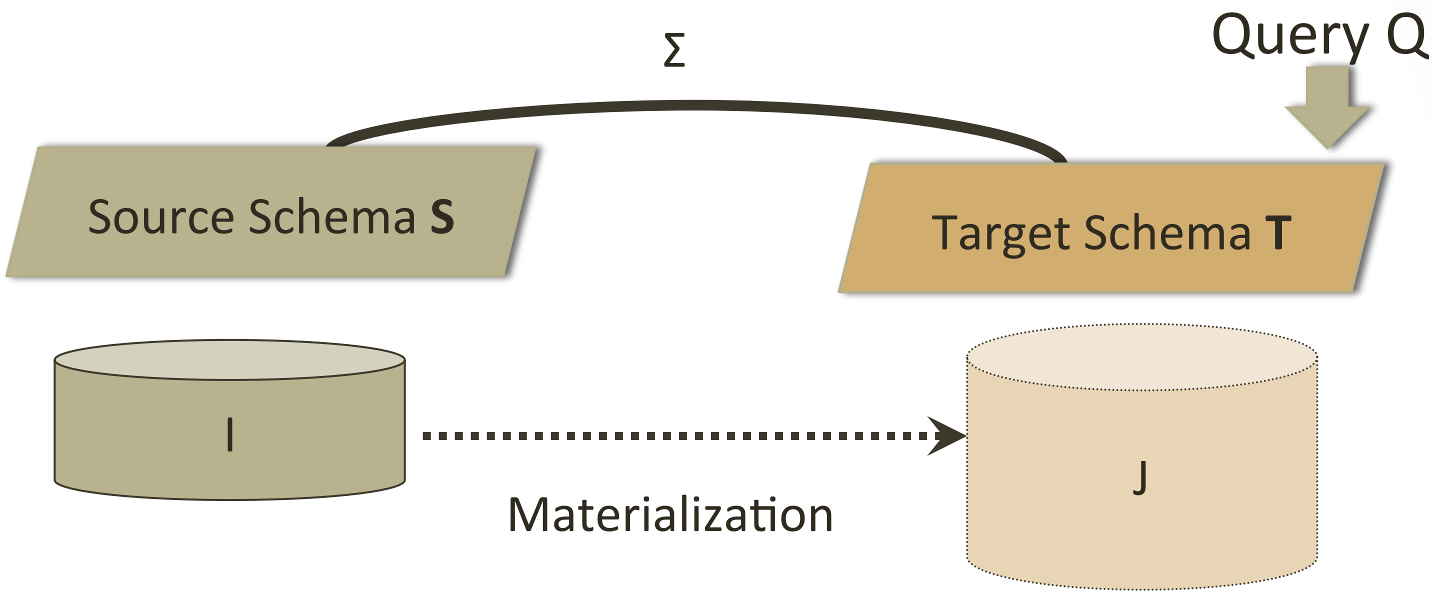
Data Integration



Data Sources

Query heterogeneous data in different sources via a *global schema*.

Data Exchange



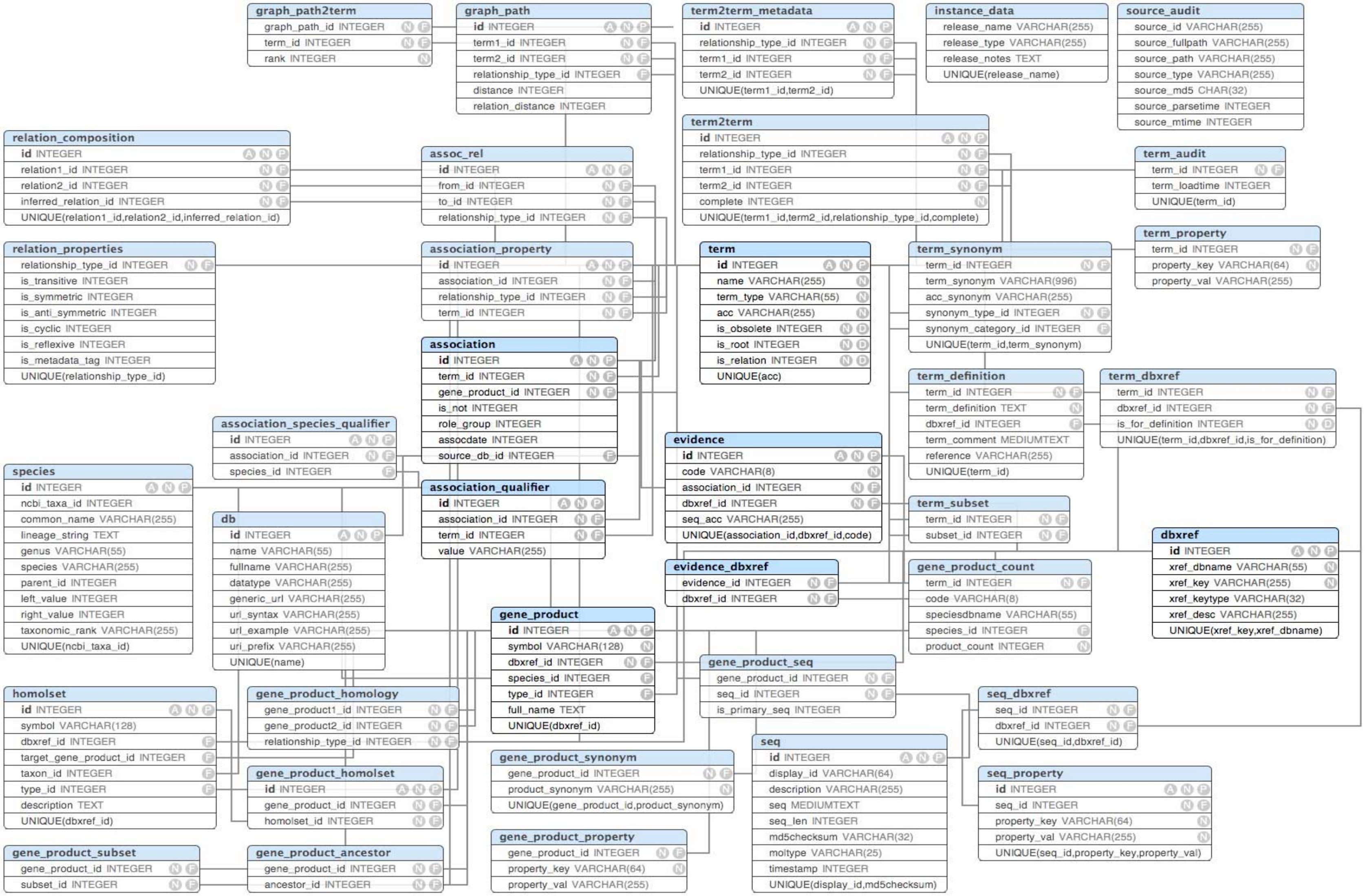
- Transform data structured under a source schema into data structured under a different target schema.
- Query heterogeneous data in different sources via the *target schema*.

Key challenge behind Data Interoperability

- Key challenge behind data integration or data exchange: specify the *relationships between schemas*.
- The relationships between schemas are typically specified as *data transformations*.

- Data transformation: (**S**, **T**, ξ)
  - Source schema **S,** target schema **T**.
  - ξ specifies *how* an instance that conforms to **S** is to be transformed to an instance that conforms to **T**.
- Deriving a correct data transformation can be a difficult task.
  - Schemas can be large and complex.

Schemas can be large and complex



Specifying a data transformation

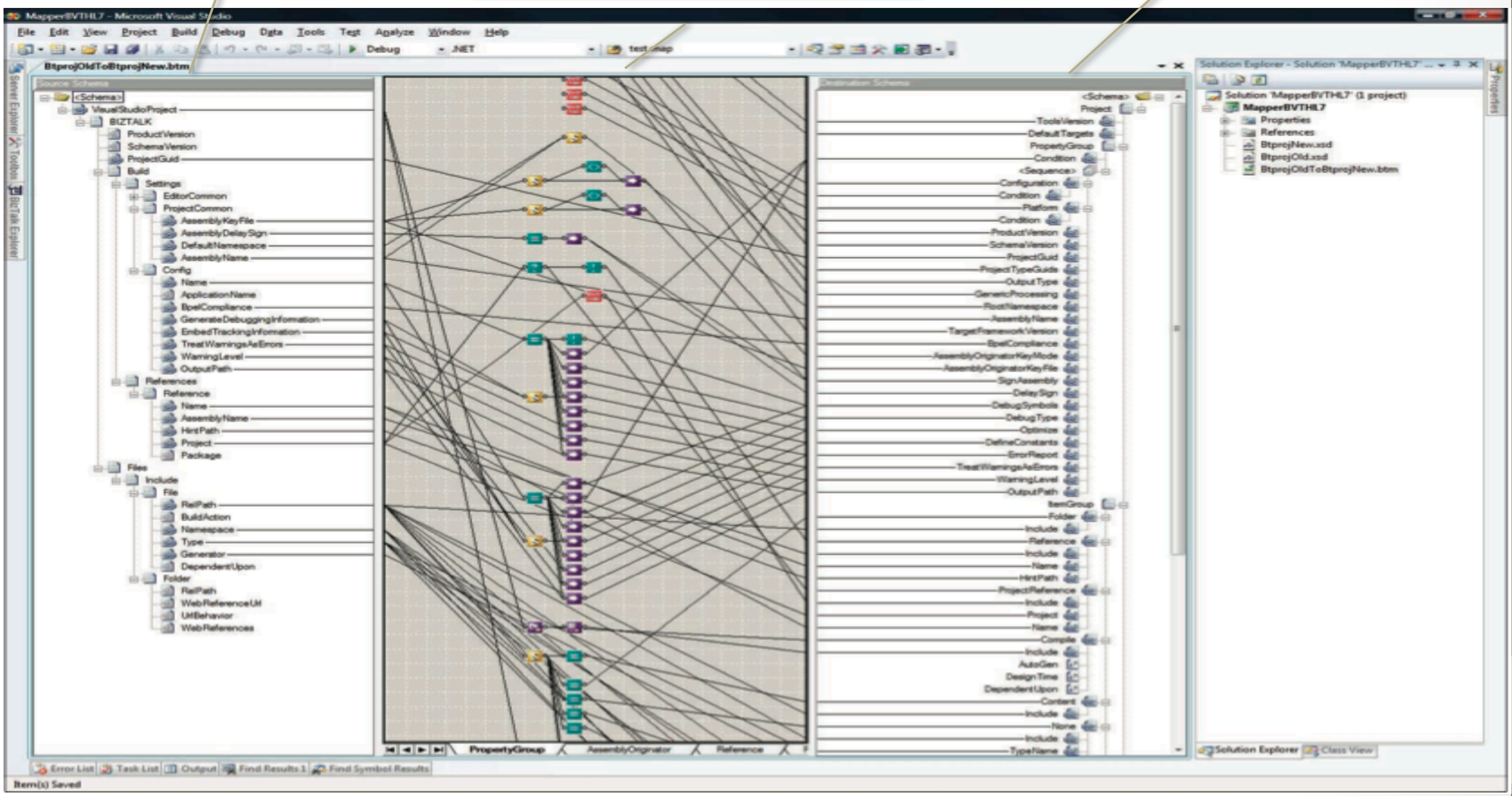
- Data transformations can be specified:
  - directly as executable code in some programming language. E.g., SQL, Java, or Pig.
    - Time-consuming, costly.
  - through a visual interface, where executable code can be generated from the visual specification.

A visual specification

Source schema

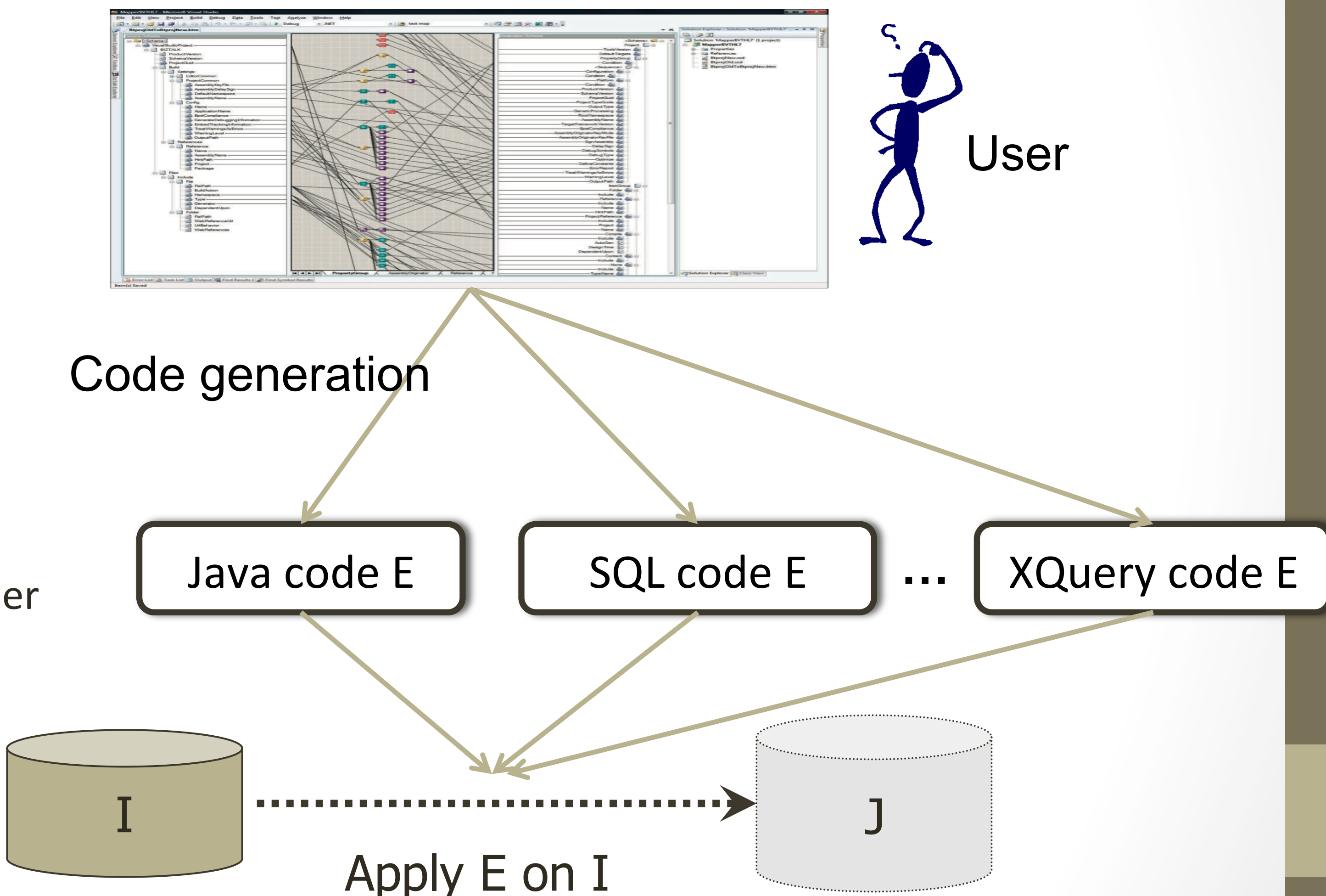
Value correspondences

Target schema



Screenshot from Bernstein and Haas 2008 CACM article. “*Information Integration in the Enterprise*”

Basic architecture behind "mapping systems"



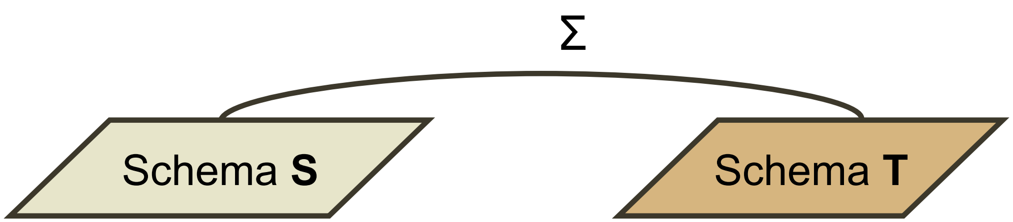
Altova Mapforce
Stylus Studio
MS Biztalk Mapper

Problems?

- (Generated) executable code of different runtime platforms tends to be complex and difficult to reason about.

- Need for higher-level abstraction of data transformations.
  - Independent of different runtime platforms.
  - Specify *what* is the relationship between the source and target schema instead of *how* data is transformed from the source to the target.

Schema Mapping



- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - *High-level, declarative assertions* Σ that specify the relationship between **S** and **T**.
  - Typically, Σ is a finite set of formulas in some suitable logical formalism (*much more on this later*).
- Schema mappings are the essential building blocks in formalizing data integration and data exchange.

Schema-Mapping Specification Languages

- **Question:**
  What is a good language for specifying schema mappings?

- **Preliminary Attempt:**
  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: $\forall x,y,z\ (P(x,y,z) \rightarrow R(x,y) \wedge T(y,z))$
- Join: $\forall x,y,z (E(x,z) \wedge F(z,y) \rightarrow R(x,z,y))$
- Combinations of the above (e.g., join + column augmentation + …)
  - $\forall x,y,z (E(x,z) \wedge F(z,y) \rightarrow \exists w\ (R(x,y) \wedge T(x,y,z,w)))$

Language for specifying Schema Mappings

All preceding tasks can be specified using:

Source-to-target tuple generating dependencies **(s-t tgds)**:

$\forall \mathbf{x}\ (\phi_S(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi_T(\mathbf{x},\mathbf{y}))$

- $\phi_S(\mathbf{x})$ is a conjunction of atoms over the source schema.
- $\psi_T(\mathbf{x},\mathbf{y})$ is a conjunction of atoms over the target schema.

Example

S = Student(studentid), Enrolls(studentid, courseid)

T = Grade(studentid, courseid, grade),

Teaches(instructerid, courseid)

$\forall s \forall c\ \text{Student}(s) \wedge \text{Enrolls}(s,c) \rightarrow \exists g\ \text{Grade}(s,c,g))$

$\forall s \forall c\ \text{Student}(s) \wedge \text{Enrolls}(s,c) \rightarrow \exists t\ \exists g\ (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g)))$

We omit all universal quantifiers for the rest of this talk.

s-t tgds

- Widely used for relational schema mappings in data exchange and data integration.

- s-t tgds are also known as *Global-Local-As-View (GLAV)* constraints. They contain:
  - Local-As-View (LAV) constraints
  - Global-As-View (GAV) constraints

  as special cases.

GLAV, GAV, LAV Schema Mappings

GAV mappings: $\phi_S(\mathbf{x}) \rightarrow R(\mathbf{x})$

- $\phi_S(\mathbf{x})$ is a conjunction of atoms over the source schema.
- R(x) is an atom of the target schema.

Example: Copy, Projection, Join, …

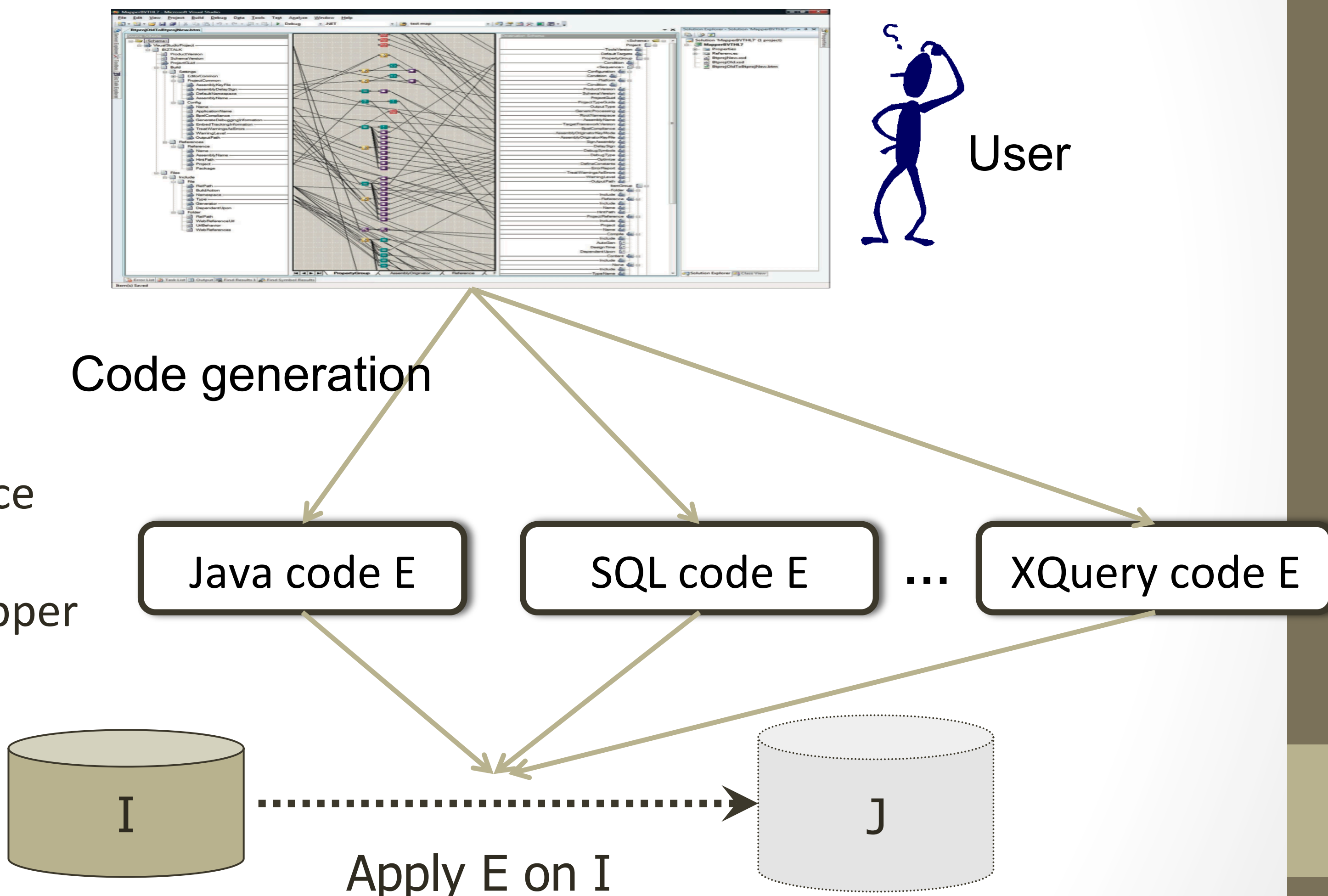
LAV mappings: $R(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi_T(\mathbf{x},\mathbf{y})$

- R(x) is an atom of the source schema.
- $\psi_T(\mathbf{x},\mathbf{y})$ is a conjunction of atoms over the target schema.

Example: Copy, Decomposition, Add an attribute to a relation …

Basic architecture behind "mapping systems"

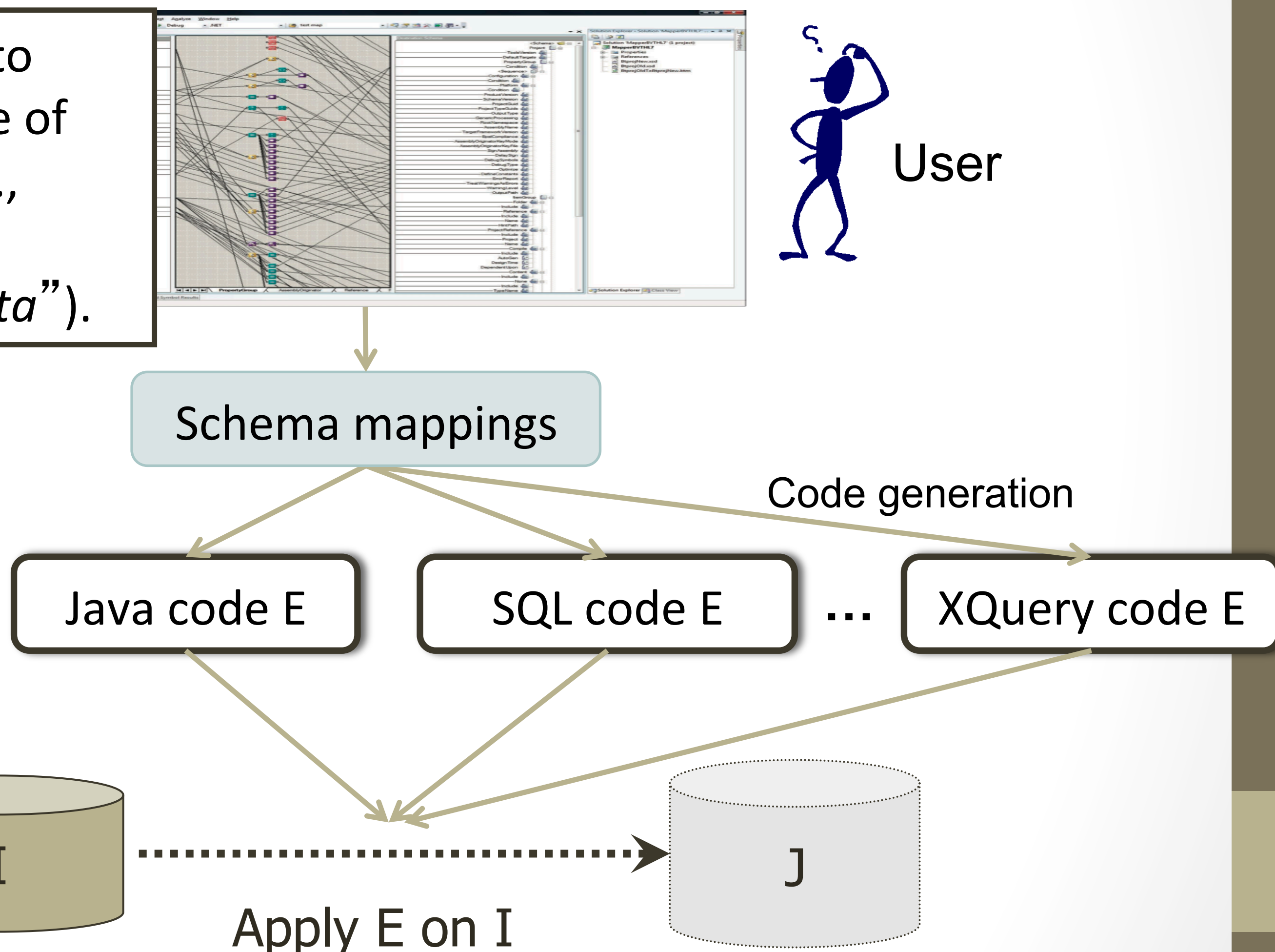
Altova Mapforce
Stylus Studio
MS Biztalk Mapper



Basic architecture behind schema-mapping design systems

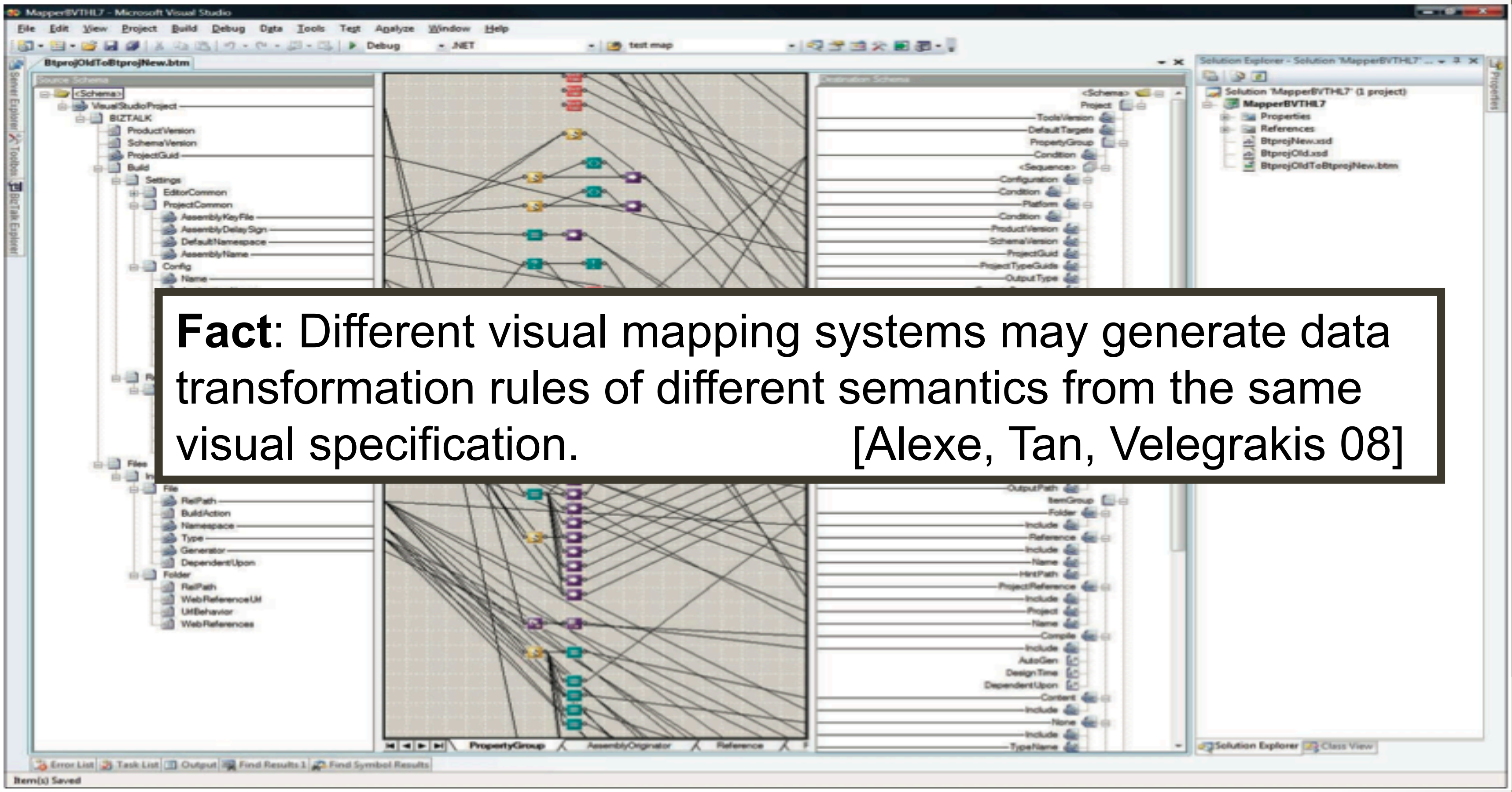
Extension of s-t tgds to handle data exchange of hierarchical data (e.g., Popa *et al*. 2002 "*Translating Web Data*").

Clio
HepTox
Spicy++



A visual specification

- How can we understand what gets generated from this?



Fact: Different visual mapping systems may generate data transformation rules of different semantics from the same visual specification. [Alexe, Tan, Velegrakis 08]

Schema Mappings

- To understand the precise semantics of what gets generated, the user will have to inspect the generated schema mapping or executable code.

- However, schema mappings and executable code can be **complex** …

Schema Mappings (one of several pages)

```
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.membership=tm2x1.organization.id,
        satisf  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.membership=tm3x6.organization.id, tm3x5.province.id=tm3x1.located.province,
                        tm2x0.country.id=tm3x1.located.country,
                satisf  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.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_26, 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.membership=tm4x6.organization.id, tm4x5.province.id=tm4x1.located.province,
                        tm2x0.country.id=tm4x1.located.country,
                satisf  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.membership=tm5x6.organization.id, tm5x5.province.id=tm5x1.located.province,
                        tm2x0.country.id=tm5x1.located.country,
                satisf  sm2x0.COUNTRY.AREA=tm5x4.country.area, sm2x0.COUNTRY.CAPITAL=tm5x4.country.capital,
                        sm2x0.COUNTRY.CODE=tm5x4.country.id, sm2x0.COUNTRY.NAME=tm5x4.country.name,
                        sm2x0.COUNTRY.POPULATION=tm5x4.country.population, sm5x1.SEA.DEPTH=tm5x0.sea.depth,
                        sm5x0.GEO_SEA.COUNTRY=tm5x1.located.country, sm5x0.GEO_SEA.PROVINCE=tm5x1.located.province,
                        sm5x1.SEA.NAME=tm5x0.sea.name ),(
```

Schema mappings can be complex

- Additional tools are needed (beyond the inspection of the visual specification and code) to design, understand, and refine schema mappings.
- **Idea:** Use “**good**” data examples.
  - Analogous to using **test cases** in understanding/debugging programs.
  - Earlier work by the database community includes:
    - Yan, Miller, Haas, Fagin – 2001
      “*Understanding and Refinement of Schema Mappings*”
    - Gottlob, Senellart – 2008
      “*Schema mapping discovery from data instances*”
    - Olston, Chopra, Srivastava – 2009
      “*Generating Example Data for Dataflow Programs*”.

The rest of this tutorial

Schema Mappings and Data Examples:

- Develop a framework for the systematic use of data examples for designing schema mappings.

- Understand both the **capabilities** and **limitations** of data examples in capturing, deriving, and designing schema mappings.

Roadmap for tutorial

First half of tutorial:

✔ • Background and Motivation
- Semantics of Schema Mappings
- From Schema Mappings to Data Examples

Second half of tutorial:

- From Data Examples to Schema Mappings
  - The Eirene and Muse Systems
  - Gottlob and Senellart's framework for discovering schema mappings
  - Learning schema mappings

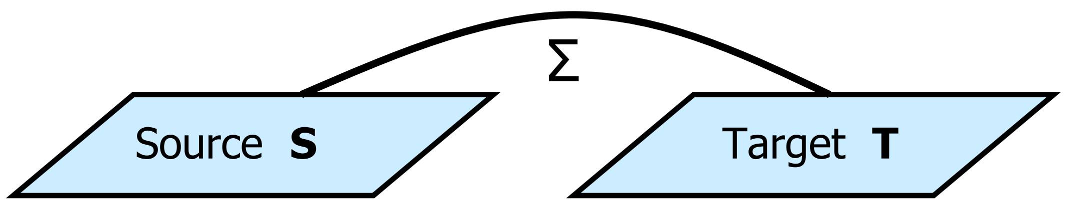
Schema Mappings
and
Data Examples

EDBT 2013 Tutorial

Genoa, Italy

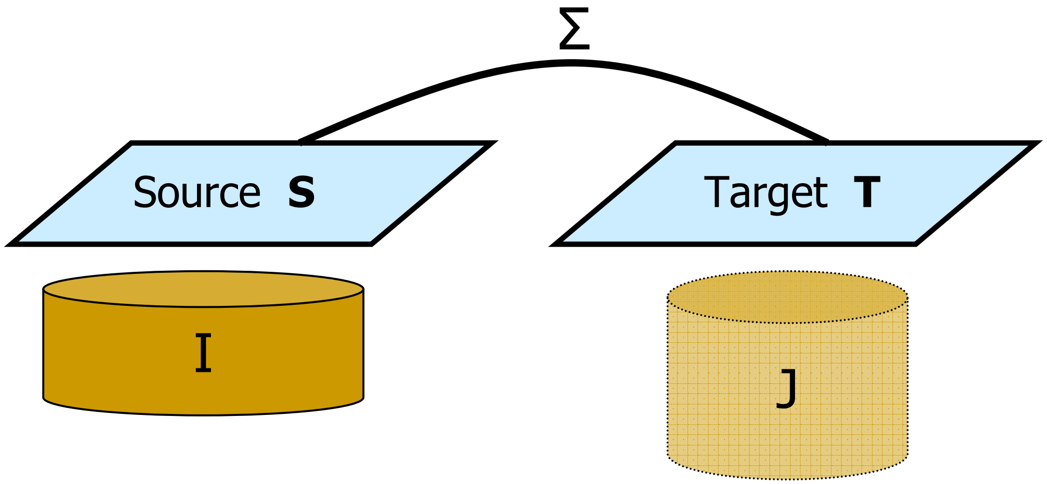
March 21, 2013

Schema Mappings



- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - High-level, declarative constraints Σ that specify the relationship between **S** and **T**.
- GLAV Schema Mapping **M** = (**S**, **T**, Σ)
  - Σ is a finite set of GLAV constraints (s-t tgds)
- GAV and LAV Schema Mappings defined in a similar way.

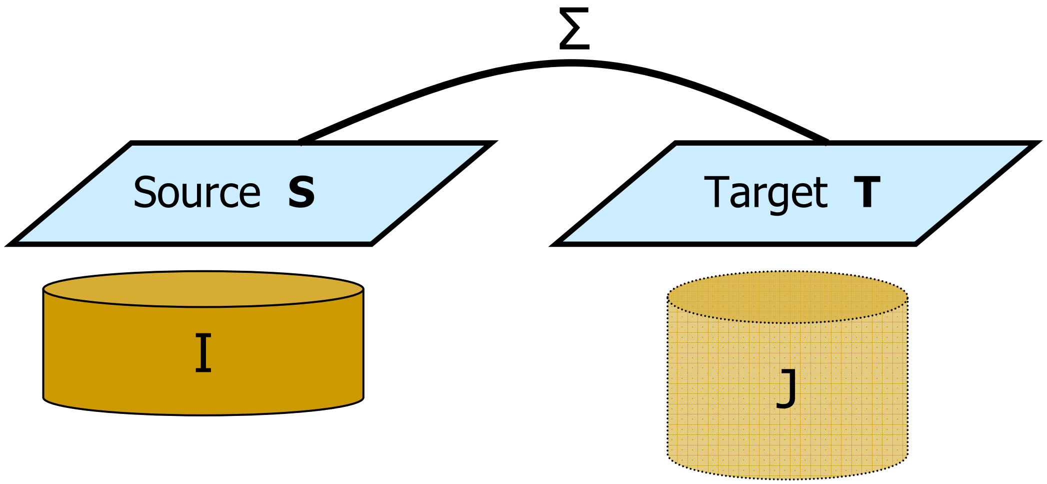
Semantics of Schema Mappings



M = (**S**, **T**, Σ) a GLAV schema mapping.

- Such a schema mapping **M** is a syntactic object.
- From a semantic point of view, **M** can be identified with the set of all positive data examples for **M**, i.e., all data examples that satisfy (the constraints of) **M**.

Data Examples



M = (**S**, **T**, Σ) a GLAV schema mapping

- Data Example: A pair (I,J) where I is a source instance and J is a target instance.
- Positive Data Example for **M**:
  - A data example (I,J) that satisfies Σ, i.e., (I,J) ⊨ Σ
  - In this case, we say that J is a solution for I w.r.t. **M.**

Data Examples

- Consider the schema mapping **M** = ({E}, {F}, Σ), where
  $\Sigma = \{\ E(x,y) \rightarrow \exists z\ (F(x,z) \wedge F(z,y))\ \}$
- Positive Data Examples (I,J) (J a solution for I w.r.t. **M**)
  - I = { E(1,2) } J = { F(1,3), F(3,2) }
  - I = { E(1,2) } J = { F(1,X), F(X,2) }
  - I = { E(1,2) } J = { F(1,3), F(3,2), F(3,4) }
  - I = { E(1,2), E(3,4) } J = { F(1,3), F(3,2), F(3,Y), F(Y,4) }

    X and Y are labelled nulls
- Negative Data Examples (I,J) (J not a solution for I w.r.t. **M**)
  - I = { E(1,2) } J = { F(1,3) }
  - I = { E(1,2) } J = { F(1,3), F(4,2) }

Schema Mappings and Data Examples

- $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ GLAV schema mapping
- $\text{Sem}(\mathbf{M}) = \{ (I,J): (I,J)$ is a positive data example for $\mathbf{M} \}$

Fact: Sem(**M**) is an infinite set

Reason:

If (I,J) is a positive data example for **M** and if $J \subseteq J'$,
then (I,J') is a positive data example for **M**.

Question:

Can **M** be "characterized" using finitely many data examples?

Goals

- Formalize what it means for a schema mapping to be "characterized" using finitely many data examples.

- Obtain technical results that shed light on both the capabilities and limitations of data examples in characterizing schema mappings.

Types of Data Examples

M = (**S**, **T**, Σ) a GLAV schema mapping

So far, we have encountered two types of examples:

- Positive Data Example:

  A data example (I,J) such that (I,J) satisfies Σ, i.e., a J is a solution for I w.r.t. **M**.

- Negative Data Example:

  A data example (I,J) such that (I,J) does not satisfy Σ, i.e., J is not a solution for I w.r.t. **M**.

A third type of example will play an important role here:

- Universal Data Example:

  A data example (I,J) such that J is a universal solution for I w.r.t. **M**.

Universal Solutions

Definition: $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ schema mapping, I source instance.

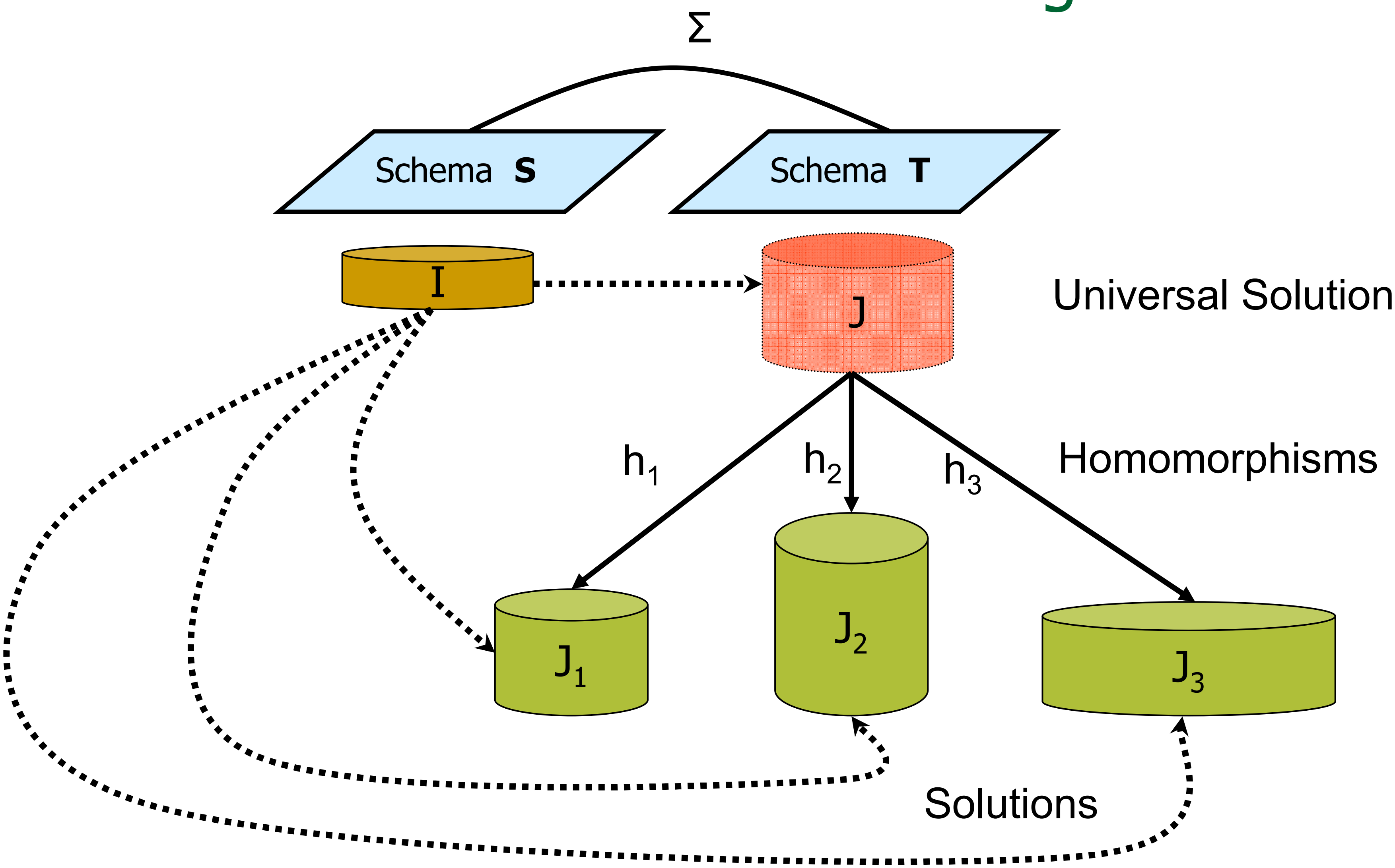
A target instance J is a universal solution for I w.r.t. M if

- J is a solution for I w.r.t. **M**.
- If J' is a solution for I w.r.t. M, then there is a homomorphism h: $J \rightarrow J'$ that is constant on adom(I), which means that:
  - If $P(a_1, \ldots, a_k) \in J$, then $P(h(a_1), \ldots h(a_k)) \in J'$

    (h preserves facts)
  - $h(c)=c$, for $c \in adom(I)$.

Note: Intuitively, a universal solution for I is a most general (= least specific) solution for I.

Universal Solutions in Data Exchange



Universal Solutions and Examples

- Consider the schema mapping **M** = ({E}, {F}, Σ), where
  $\Sigma = \{ E(x,y) \rightarrow \exists z\, (F(x,z) \wedge F(z,y)) \}$

- Source instance I = { E(1,2) }

- Solutions for I :
  - J_1 = { F(1,2), F(2,2) }
  - J_2 = { F(1,X), F(X,2) }
  - J_3 = { F(1,X), F(X,2), F(1,Y), F(Y,2) }
  - J_4 = { F(1,X), F(X,2), F(3,3) }

  (where X and Y are labeled null values)
  - ...

Data Examples:

- (I,J_1) positive, **not** universal
- (I,J_2) universal (and positive)
- (I,J_3) universal (and positive)
- (I,J_4) positive, **not** universal

Universal Solutions and Schema Mappings

Note: A key property of GLAV schema mappings is the **existence of universal solutions**.

Theorem (FKMP 2003) **M** = (**S**, **T**, Σ) a GLAV schema mapping.

- Every source instance I has a universal solution J w.r.t. M,
- Moreover, the chase procedure can be used to construct, given a source instance I, a canonical universal solution $\text{chase}_{\mathbf{M}}(I)$ for I in polynomial time.

Note: Universal solutions have become the preferred semantics in data exchange (the preferred solutions to materialize).

The Chase Procedure

Chase Procedure for GLAV **M** = (**S**, **T**, Σ): Given a source instance I, build a target instance $\text{chase}_{\mathbf{M}}(I)$ that satisfies every s-t tgd in Σ as follows.

Whenever the LHS of some s-t tgd in Σ evaluates to true:

- Introduce new facts in $\text{chase}_{\mathbf{M}}(I)$ as dictated by the RHS of the s-t tgd.
- In these facts, each time existential quantifiers need witnesses, introduce new variables (labeled nulls) as values.

The Chase Procedure

Example: Transforming edges to paths of length 2
M = (**S**, **T**, Σ) schema mapping with
Σ : $\forall x\, \forall y(E(x,y) \rightarrow \exists z(F(x,z) \wedge F(z,y)))$

The chase returns a relation obtained from E by adding a new node between every edge of E.

- If I = { E(1,2) }, then $chase_{\mathbf{M}}(I)$ = { F(1,X), F(X,2) }
- If I = { E(1,2), E(2,3), E(1,4) }, then
  $chase_{\mathbf{M}}(I)$ = { F(1,X), F(X,2), F(2,Y), F(Y,3), F(1,Z), F(Z,4) }

The Chase Procedure

Example : Collapsing paths of length 2 to edges

M = (**S**, **T**, Σ) GAV schema mapping with

Σ : $\forall x\, \forall y\, \forall z\, (E(x,z) \wedge E(z,y) \rightarrow F(x,y))$

- If I = { E(1,3), E(2,4), E(3,4) }, then $\text{chase}_{\mathbf{M}}(I)$ = { F(1,4) }.
- If I = { E(1,3), E(2,4), E(3,4), E(4,3) }, then $\text{chase}_{\mathbf{M}}(I)$ = { F(1,4), F(2,3), F(3,3), F(4,4) }.

Note: **No** new variables are introduced in the GAV case.

Characterizing Schema Mappings

- **M** = (**S**, **T**, Σ) GLAV schema mapping
- Sem(**M**) = { (I,J): (I,J) is a positive data example for **M** }

Question:

Can **M** be "characterized" using finitely many data examples?

More formally, this asks:

Is there is a finite set ***D*** of data examples such that **M** is the only (up to logical equivalence) schema mapping for which every example in ***D*** is of the same type as it is for **M**?

Warm-up: The Copy Schema Mapping

Let **M** be the binary copy schema mapping specified by the constraint

$$\forall x\ \forall y\ (E(x,y) \rightarrow F(x,y)).$$

Question: Which is the "most representative" data example for **M**, hence a good candidate for "characterizing" it?

Intuitive Answer: (I_1,J_1) with $I_1 = \{ E(a,b) \}$, $J_1 = \{ F(a,b) \}$

Facts: It will turn out that:

- (I_1,J_1) "characterizes" **M** among all LAV schema mappings.
- (I_1,J_1) does not "characterize" **M** among all GLAV schema mappings; in fact, not even among all GAV schema mappings.

Reason: (I_1,J_1) is also a universal example for the GAV schema mapping specified by $\forall x\ \forall y\ \forall u\ \forall v\ (E(x,y) \wedge E(u,v) \rightarrow F(x,v))$.

Notions of Unique Characterizability

Definition: **M** = (**S**, **T**, Σ) a GLAV schema mapping, ***C*** a class of GLAV constraints.

- Let **P** and **N** be two finite sets of positive and negative examples for **M**. We say that **P** and **N** uniquely characterize **M** w.r.t. ***C*** if for every finite set $\Sigma' \subseteq$ ***C*** such that **P** and **N** are sets of positive and negative examples for **M'** = (**S**, **T**, Σ'), we have that $\Sigma \equiv \Sigma'$.

- Let **U** be a finite set of universal examples for **M**. We say that **U** uniquely characterizes **M** w.r.t. ***C*** if for every finite set $\Sigma' \subseteq$ ***C*** such that **U** is a set of universal examples for **M'** = (**S**, **T**, Σ'), we have that $\Sigma \equiv \Sigma'$.

Relationships between Unique Characterizability Notions

Proposition: **M** = (**S**, **T**, Σ) a GLAV schema mapping, ***C*** a class of GLAV constraints.

If **M** is uniquely characterizable w.r.t. ***C*** by two finite sets of positive and negative examples, then **M** is also uniquely characterizable w.r.t. ***C*** by a finite set of universal examples.

Proof Idea: Uniquely characterizing

positive examples: (I^+_1, J^+_1), (I^+_2, J^+_2), … and

negative examples: (I^-_1, J^-_1), (I^-_2, J^-_2), …

give rise to uniquely characterizing

universal examples: $(I^+_1, \mathrm{chase}_{\mathbf{M}}(I^+_1))$, $(I^+_2, \mathrm{chase}_{\mathbf{M}}(I^+_2))$, …

$(I^-_1, \mathrm{chase}_{\mathbf{M}}(I^-_1))$, $(I^+_2, \mathrm{chase}_{\mathbf{M}}(I^+_2))$, …

Relationships between Unique Characterizability Notions

- So, unique characterizability via positive and negative examples implies unique characterizability via universal examples.

- The converse, however, is not always true.

- For this reason, we will focus on unique characterizability via universal examples.

Unique Characterizations via Universal Examples

Reminder -

Definition: Let **M** = (**S**, **T**, Σ) be a GLAV schema mapping.

- A **universal example** for **M** is a data example (I,J) such that J is a universal solution for I w.r.t. **M**.
- Let **U** be a finite set of universal examples for **M**, and let ***C*** be a class of GLAV constraints.

  We say that **U** **uniquely characterizes** **M** **w.r.t.** ***C*** if for every finite set $\Sigma' \subseteq$ ***C*** such that **U** is a set of universal examples for the schema mapping **M'** = (**S**, **T**, Σ'), we have that $\Sigma \equiv \Sigma'$.

Unique Characterizations via Universal Examples

Question:

Which GLAV schema mappings can be uniquely characterized by a finite set of universal examples and w.r.t. to what classes of constraints?

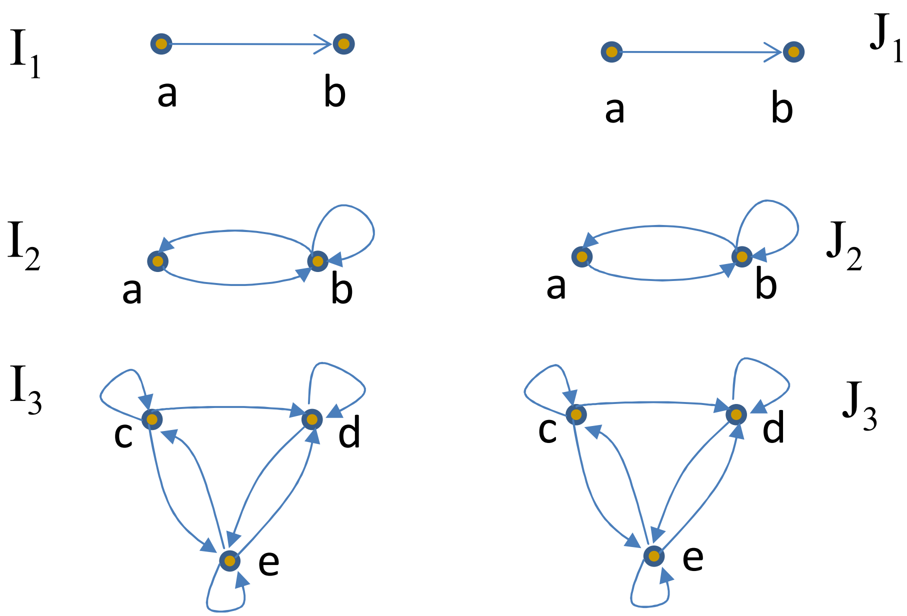
Unique Characterizations Warm-Up

Theorem: Let **M** be the binary copy schema mapping specified by the constraint $\forall x \, \forall y \, (E(x,y) \rightarrow F(x,y))$.

- The set **U** = { (I_1, J_1) } with I_1 = { E(a,b }, J_1 = { F(a,b) } uniquely characterizes **M** w.r.t. the class of all LAV constraints.
- There is a finite set **U'** consisting of three universal examples that uniquely characterizes **M** w.r.t. the class of all GAV constraints.
- There is **no** finite set of universal examples that uniquely characterizes **M** w.r.t. the class of all GLAV constraints.

Unique Characterizations Warm-Up

The set **U'** = { (I_1,J_1), (I_2,J_2), (I_3,J_3) } uniquely characterizes the copy schema mapping w.r.t. to the class of all GAV constraints.



Unique Characterizations of LAV Mappings

Theorem: If **M** = (**S**, **T**, Σ) is a LAV schema mapping, then there is a finite set **U** of universal examples that uniquely characterizes **M** w.r.t. the class of all LAV constraints.

Hint of Proof:

- Let $d_1, d_2, \ldots, d_k$ be k distinct elements, where k = maximum arity of the relations in **S**.
- **U** consists of all universal examples (I, J) with $I = \{ R(c_1,\ldots,c_m) \}$ and $J = chase_{\mathbf{M}}(\{ R(c_1,\ldots,c_m) \})$, where each c_i is one of the d_j's.

Illustration of Unique Characterizability

Let **M** be the binary projection schema mapping specified by

$$\forall x\ \forall y\ (P(x,y) \ \rightarrow Q(x))$$

- The following set **U** of universal examples uniquely characterizes **M** w.r.t. the class of all LAV constraints:

  U = $\{ (I_1, J_1),\ (I_2, J_2) \}$, where

  - $I_1 = \{ P(c_1,c_2) \}$, $J_1 = \{ Q(c_1) \}$
  - $I_2 = \{ P(c_1,c_1) \}$, $J_2 = \{ Q(c_1) \}$.

Illustration of Unique Characterizability

Let **M** be the schema mapping specified by

$\forall x\, \forall y\, (P(x,y) \rightarrow Q(x))$ and $\forall x\, (P(x,x) \rightarrow \exists y\, R(x,y))$

- The following set **U** of universal examples uniquely characterizes **M** w.r.t. the class of all LAV constraints:

  U = $\{ (I_1, J_1),\ (I_2, J_2) \}$, where

  - $I_1 = \{ P(c_1,c_2) \}, \quad J_1 = \{ Q(c_1) \}$
  - $I_2 = \{ P(c_1,c_1) \}, \quad J_2 = \{ Q(c_1), R(c_1,Y) \}.$

Number of Uniquely Characterizing Examples

Note:

- The number of universal examples needed to uniquely characterize a LAV schema mapping is bounded by an exponential in the maximum arity of the relations in the source schema.
- This bound turns out to be tight.

Theorem: For $n \geq 3$, let $\mathbf{M}_n$ be the n-ary copy schema mapping specified by the constraint

$$\forall x_1 \ldots \forall x_n(P(x_1,\ldots,x_n) \rightarrow Q(x_1,\ldots,x_n)).$$

If $\mathbf{U}$ is a set of universal examples that uniquely characterizes $\mathbf{M}_n$ w.r.t. the class of LAV constraints, then $|\mathbf{U}| \geq 2^n - 2$.

Unique Characterizations of GAV Mappings

Note: Recall that for the schema mapping specified by the binary copy constraint $\forall x\ \forall y\ (E(x,y) \rightarrow F(x,y))$, there is a finite set of universal examples that uniquely characterizes it w.r.t. the class of all GAV constraints.

In contrast,

Theorem: Let **M** be the GAV schema mapping specified by $\forall x\ \forall y\ \forall u\ \forall v\ \forall w\ (E(x,y) \wedge E(u,v) \wedge E(v,w) \wedge E(w,u) \rightarrow F(x,y))$.
There is **no** finite set of universal examples that uniquely characterizes **M** w.r.t. the class of all GAV constraints.

Unique Characterizations of GAV Mappings

Theorem: Let **M** be the GAV schema mapping specified by
$\forall x \forall y \forall u \forall v \forall w\, (E(x,y) \wedge E(u,v) \wedge E(v,w) \wedge E(w,u) \rightarrow F(x,y))$.
There is **no** finite set of universal examples that uniquely characterizes **M** w.r.t. the class of all GAV constraints.

Note:

- Extends to every GAV schema mapping specified by
  $\forall x \forall y\, (E(x,y) \wedge Q_G \rightarrow F(x,y))$, where Q_G is the canonical conjunctive query of a graph G containing a cycle.
  This will be a consequence of more general results to be discussed in what follows.

(Non)-Characterizable GAV Schema Mappings

In summary, we have that

- $\forall x\, \forall y\, (E(x,y) \rightarrow F(x,y))$

  is uniquely characterizable by finitely many (in fact, three) universal examples w.r.t. the class of all GAV constraints.

- $\forall x\, \forall y\, \forall u\, \forall v\, \forall w\, (E(x,y) \wedge E(u,v) \wedge E(v,w) \wedge E(w,u) \rightarrow F(x,y))$

  is not uniquely characterizable by finitely many universal examples w.r.t. the class of all GAV constraints.

Question: How can this difference be explained?

Characterizing GAV Schema Mappings

- **Question:**
  - What is the reason that some GAV schema mappings **are** uniquely characterizable w.r.t. the class of all GAV constraints while some others are **not**?
  - Is there an algorithm for deciding whether or not a given GAV schema mapping is uniquely characterizable w.r.t. the class of all GAV constraints?

- **Answer:**
  - The answers to these questions are closely connected to database constraints and **homomorphism dualities**.

Homomorphisms

Notation: **A**, **B** relational structures (e.g., graphs)

- **A** $\rightarrow$ **B** means there is a **homomorphism** h from **A** to **B**, i.e., a function h from the universe of **A** to the universe of **B** such that if $P(a_1,\ldots,a_m)$ is a fact of **A**, then $P(h(a_1), \ldots, h(a_m))$ is a fact of **B**.
  - **Example:** **G** $\rightarrow$ $\mathbf{K_2}$ if and only if **G** is 2-colorable

- $\rightarrow\mathbf{A} = \{\mathbf{B} : \mathbf{B} \rightarrow \mathbf{A}\}$
  - **Example:** $\rightarrow\mathbf{K_2}$ = Class of 2-colorable graphs

- $\mathbf{A}\rightarrow = \{\mathbf{B} : \mathbf{A} \rightarrow \mathbf{B}\}$
  - **Example:** $\mathbf{K_2}\rightarrow$ = Class of graphs with at least one edge.

Homomorphism Dualities

- **Definition:** Let **D** and **F** be two relational structures
  - (**F**,**D**) is a **duality pair** if for every structure **A**

    A $\rightarrow$ **D** if and only if (**F** $\not\rightarrow$ **A**).

    In symbols, $\rightarrow$**D** = **F**$\not\rightarrow$
  - In this case, we say that **F** is an **obstruction** for **D**.

- **Examples:**
  - For graphs, ($\mathbf{K_2}$, $\mathbf{K_1}$) is a duality pair, since

    $\mathbf{G} \rightarrow \mathbf{K_1}$ if and only if $\mathbf{K_2} \not\rightarrow \mathbf{G}$.
  - **Gallai-Hasse-Roy-Vitaver Theorem (~1965)** for directed graphs
    Let $\mathbf{T_k}$ be the linear order with k elements, $\mathbf{P_{k+1}}$ be the path with k+1 elements. Then ($\mathbf{P_{k+1}}$, $\mathbf{T_k}$) is a duality pair, since for every **H**

    $\mathbf{H} \rightarrow \mathbf{T_k}$ if and only if $\mathbf{P_{k+1}} \not\rightarrow \mathbf{H}$.

Homomorphism Dualities

- **Theorem** (**König 1936**): A graph is 2-colorable if and only if it contains no cycle of odd length.

  In symbols, $\rightarrow \mathbf{K}_2 = \bigcap_{i\geq 0} (\mathbf{C}_{2i+1} \nrightarrow)$.

- **Definition**: Let ***F*** and ***D*** be two sets of structures. We say that (***F***, ***D***) is a **duality pair** if for every structure **A**, TFAE
  - There is a structure **D** in ***D*** such that $\mathbf{A} \rightarrow \mathbf{D}$.
  - For every structure **F** in ***F***, we have $\mathbf{F} \nrightarrow \mathbf{A}$.

  In symbols, $\bigcup_{\mathbf{D} \in \boldsymbol{D}} (\rightarrow \mathbf{D}) = \bigcap_{\mathbf{F} \in \boldsymbol{F}} (\mathbf{F} \nrightarrow)$.

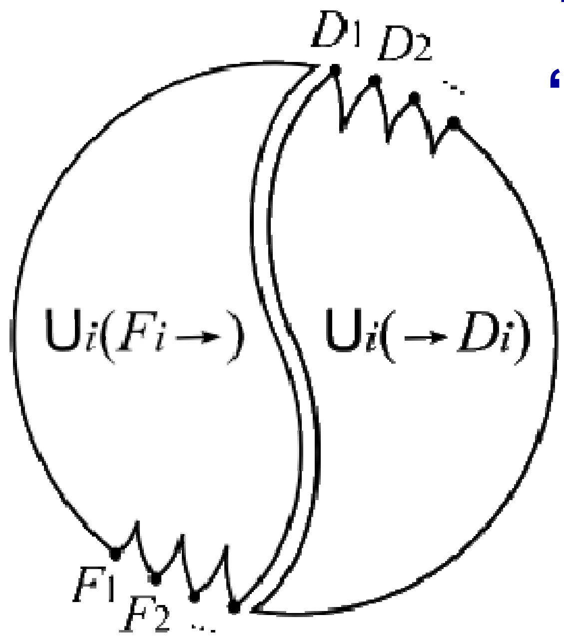
  In this case, we say that ***F*** is an **obstruction set** for ***D***.

Homomorphism Dualities

Duality Pair ($\boldsymbol{F}$,$\boldsymbol{D}$),where

$\boldsymbol{F}$ = {F_1,F_2,…}

$\boldsymbol{D}$ = {D_1,D_2,…}



The Yin

"Dreams": $\cup_i$ ($\rightarrow D_i$)

The Yang

"Fears": $\cup_i$ ($F_i \rightarrow$)

Unique Characterizations and Homomorphism Dualities

Theorem: Let **M** = (**S**, **T**, Σ) be a GAV mapping.
Then the following statements are equivalent:

- **M** is uniquely characterizable via universal examples w.r.t. the class of all GAV constraints.
- For every target relation symbol R, the set ***F***(**M**,R) of the **canonical structures** of the GAV constraints in Σ with R as their head is the obstruction set of some finite set ***D*** of structures.

Canonical Structures of GAV Constraints

Definition:

- The **canonical structure** of a GAV constraint

  $$\forall x\, (\varphi_1(x) \wedge \ldots \wedge \varphi_\kappa(x) \rightarrow R(x_{i_1},\ldots,x_{i_m}))$$

  is the structure consisting of the atomic facts $\varphi_1(x), \ldots, \varphi_\kappa(x)$ and having constant symbols $c_1,\ldots,c_m$ interpreted by the variables $x_{i_1},\ldots,x_{i_m}$ in the atom $R(x_{i_1},\ldots,x_{i_m})$.

- Let $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a GAV schema mapping.

  For every relation symbol R in $\mathbf{T}$, let $\boldsymbol{F}(\mathbf{M},R)$ be the set of all canonical structures of GAV constraints in Σ with the target relation symbol R in their head.

Canonical Structures

Examples:

- GAV constraint σ

  $\forall x\ \forall y\ \forall z\ (E(x,y) \wedge E(y,z) \rightarrow F(x,z))$

  - Canonical structure: $\mathbf{A}_{\sigma}$ = ({x,y,z}, {(E(x,y),E(y,z)},x,z)
  - Constants c_1 and c_2 interpreted by the distinguished elements x and z.

- GAV constraint θ

  $\forall x\ \forall y\ \forall z(E(x,y) \wedge E(y,z) \rightarrow F(x,x))$

  - Canonical structure: $\mathbf{A}_{\tau}$ = ({x,y,z}, {E(x,y),E(y,z)},x,x)
  - Constants c_1 and c_2 both interpreted by the distinguished element x.

Unique Characterizations and Homomorphism Dualities

Theorem: Let **M** = (**S**, **T**, Σ) be a GAV mapping.
Then the following statements are equivalent:

- **M** is uniquely characterizable via universal examples w.r.t. the class of all GAV constraints.
- For every target relation symbol R, the set ***F***(**M**,R) of the **canonical structures** of the GAV constraints in Σ with R as their head is the obstruction set of some finite set ***D*** of structures.

Illustration

Let **M** be the GAV schema mapping specified by

$$\forall x\, (R(x,x) \rightarrow P(x)).$$

- Canonical structure F = ({x}, {R(x,x)}, x)
- Consider D = ({a,b}, {R(a,b), R(b,a), R(b,b)}, a})

Fact: (F,D) is a duality pair, because it is easy to see that for every structure G=(V,R,d), we have that

$$G \rightarrow D \text{ if and only if } F \nrightarrow G.$$

Consequently, **M** is uniquely characterizable via universal examples w.r.t. the class of all GAV constraints.

Unique Characterizations and Homomorphism Dualities

Question:

- Is there an algorithm to decide when a GAV mapping is uniquely characterizable via a finite set of universal examples w.r.t. to the class of all GAV constraints?

- If so, what is the complexity of this decision problem?

c-Acyclicity

Definition: Let $\mathbf{A} = (A, R_1,\ldots,R_m,c_1,\ldots c_k)$ be a relational structure with constants $c_1,\ldots,c_k$.

- The **incidence graph** inc(**A**) of **A** is the bipartite graph with
  - nodes the elements of A and the facts of A
  - edges between elements and facts in which they occur
- The structure **A** is **c-acyclic** if
  - Every cycle of Inc(A) contains at least one constant c_i, and
  - Only constants may occur more than once in the same fact.

Example:

- **A** = ({1,2,3}, {R((1,2,3), Q(1,2)}, 1) is c-acyclic
  - the cycle 1 , R(1,2,3) , 2, Q(1,2), 1 contains the constant 1, and it is the only cycle of inc(**A**).
- **A** = ({1,2,3}, {R((1,2,3), Q(1,2)}, 3) is not c-acyclic
  - the cycle 1 , R(1,2,3) , 2, Q(1,2), 1 contains no constant.

When do Homomorphism Dualities Exist?

Theorem:
Let ***F*** be a finite set of relational structures with constants consisting of homomorphically incomparable core structures.

- The following statements are equivalent:
  - ***F*** is an **obstruction set** of some finite set ***D*** of structures.
  - Each structure **F** in ***F*** is **c-acyclic**.

- Moreover, there is an algorithm that, given such a set ***F*** consisting of c-acyclic structures, computes a finite set ***D*** of structures such that (***F***, ***D***) is a duality pair.

Note: Extends results of Foniok, Nešetřil, and Tardif – 2008.

Normal Forms

Definition: A GAV schema mapping is in normal form if for every target relation symbol R, the set $\boldsymbol{F}$(**M**,R) of the canonical structures of the GAV constraints in Σ with R as their head consists of homomorphically incomparable cores.

Fact:

- Every GAV schema mapping is logically equivalent to a GAV schema mapping in normal form.
- There is an algorithm based on conjunctive-query containment that transforms a given GAV schema mapping to a GAV schema mapping in normal form.

Unique Characterizations and Homomorphism Dualities

Theorem: Let **M** = (**S**, **T**, Σ) be a GAV schema mapping in normal form. Then the following statements are equivalent:

- **M** is **uniquely characterizable via universal examples** w.r.t. the class of all GAV constraints.
- For every target relation symbol R, the set ***F***(**M**,R) is the **obstruction set** of some finite set of structures.
- For every target relation symbol R, the set ***F***(**M**,R) consists entirely of **c-acyclic** structures.

Complexity of Unique Characterizations of GAV Mappings

Theorem:

- This following problem is in LOGSPACE:

  Given a GAV mapping **M** in normal form, is it uniquely characterizable via universal examples w.r.t. the class of all GAV constraints?

- The following problem is NP-complete:

  Given a GAV mapping **M**, is it uniquely characterizable via universal examples w.r.t. the class of all GAV constraints?

Note:

- Recall that every GAV mapping can be transformed to a logically equivalent one in normal form.

Applications

- The GAV schema mapping **M** specified by

  $$\forall x \forall y (E(x,y) \rightarrow F(x,y))$$

  is uniquely characterizable (the canonical structure is c-acyclic).

- More generally, if **M** is a GAV schema mapping specified by a tgd in which all variables in the LHS are exported to the RHS, then **M** is uniquely characterizable (reason: cycles in incidence graph contain constants).

- The GAV schema mapping **M** specified by

  $$\forall x \forall y \forall u \forall v \forall w (E(x,y) \wedge E(u,v) \wedge E(v,w) \wedge E(w,u) \rightarrow F(x,y)).$$

  is **not** uniquely characterizable:

  the canonical structure contains a cycle with no constant on it, namely,

  $$u, E(u,v), v, E(v,w), w, E(w,u), u$$

- The GAV schema mapping **M** specified by

  $$\forall x \forall y \forall u (E(x,y) \wedge E(u,u) \rightarrow F(x,y))$$

  is **not** uniquely characterizable.

More Applications

- The GAV schema mapping specified by the constraint
  $$\forall x \forall y \forall z\, (E(x,y) \wedge E(y,z) \rightarrow F(x,z))$$
  is uniquely characterizable via universal examples.

- Let **M** be the GAV schema mappings specified by the constraints
  - σ: $\forall x \forall y \forall z\, (E(x,y) \wedge E(y,z) \wedge E(z,x) \rightarrow F(x,z))$
  - τ: $\forall x \forall y\, (E(x,y) \wedge E(y,x) \rightarrow F(x,x))$

  The canonical structures of these constraints are
  - A_σ = ({x,y,x} {E(x,y), E(y,z), E(z,x)}, x, z)
  - A_τ = ({x,y}, {E(x,y), E(y,x)}, x, x)

  - Both are c-acyclic; hence {A_σ, A_τ} is an obstruction set of a finite set of structures.
  - Therefore, **M** is uniquely characterizable via universal examples.

Synopsis

- Introduced and studied the notion of unique characterization of a schema mapping by a finite set of universal examples.
- Every LAV schema mapping is uniquely characterizable via universal examples w.r.t. the class of all LAV constraints.
- Necessary and sufficient condition, and an algorithmic criterion for a GAV schema mapping to be uniquely characterizable via universal examples w.r.t. the class of all GAV constraints.
  - Tight connection with homomorphism dualities.

Open Problems

- When is a LAV schema mapping uniquely characterizable by a "small" number of universal examples w.r.t. to the class of all LAV constraints?
  - Same question for GAV schema mappings.

- When is a GLAV schema mapping uniquely characterizable by finitely many universal examples w.r.t. to the class of all GLAV constraints?
  - We do not even know whether this problem is decidable.

References

- This part of the tutorial is based mainly on the paper "Characterizing Schema Mappings via Data Examples" by B. Alexe, B. ten Cate, Ph. Kolaitis, W.-C. Tan in ACM TODS 2011.
  - Earlier versions appeared in PODS 2010 and CP 2011.

- For an introduction on homomorphism dualities, see the book "Graphs and Homomorphisms" by P. Hell and J. Nešetril, Cambridge University Press 2004.

Roadmap

This tutorial is about schema mappings and data examples.

- This part of the tutorial focused on the direction
  - ***From schema mappings to data examples:***

    Given a schema mapping, how can we characterize it using finitely many "good" data examples?

- The next part of the tutorial will focus on the other direction:
  - ***From data examples to schema mappings***.

Back-up Slides

Armstrong Bases and Armstrong Databases

Definition: (Fagin - 1982; implicit in Armstrong - 1974)
Σ and ***C*** two sets of constraints over the same schema. An **Armstrong database for** Σ **w.r.t.** ***C*** is a database D such that for every $\sigma \in$ ***C***, we have that $\Sigma \models \sigma$ if and only if $D \models \sigma$.

Note: Armstrong databases were extensively studied in the context of the implication problem for database constraints.

Definition: Σ and ***C*** two sets of constraints over the same schema. An **Armstrong basis for** Σ **w.r.t.** ***C*** is a finite set **D** of databases such that for every $\sigma \in$ ***C***, we have that

$$\Sigma \models \sigma \text{ if and only if } D \models \sigma, \text{ for every } D \in \mathbf{D}.$$

Armstrong Databases vs. Armstrong Bases

Example: $\Sigma = \{ P(x) \rightarrow P'(x), Q(x) \rightarrow Q'(x) \}$

- There is **no** Armstrong database for Σ w.r.t. the class of all LAV constraints.
- There is an Armstrong basis for Σ w.r.t. the class of all LAV constraints, namely, $\mathbf{D} = \{ D_1, D_2 \}$ with

  $D_1 = \{ P(a), P'(a) \}$, $D_2 = \{ Q(a), Q'(a) \}$.

Note:

- Armstrong bases do not seem to have been studied earlier.
- Much of the earlier work on Armstrong bases focused on unirelational databases and typed constraints; in this case, an Armstrong basis exists if and only if an Armstrong database exists.

Universal Examples and Armstrong Bases

Theorem: Let **M** = (**S**, **T**, Σ) be a GLAV schema mapping, and let ***C*** be a set of GLAV constraints. The following are equivalent:

1. There is a finite set **U** of universal examples that uniquely characterizes **M** w.r.t. ***C***.
2. There is an Armstrong basis **D** for Σ w.r.t. ***C***.

Note: The above result:

- Reinforces the "goodness" of universal examples.
- Reveals an a priori unexpected connection between a key notion in data exchange and (a relaxation of) a key notion in database dependency theory.

Schema Mappings and Data Examples

Earlier part of this tutorial:

- *From schema mappings to data examples*:
  - Given a schema mapping, how can we characterize it using finitely many "good" data examples?

This part of the tutorial will focus on the other direction:

- *From data examples to schema mappings*.
  - The Eirene and Muse Systems
    - Use data examples to derive and understand schema mappings.

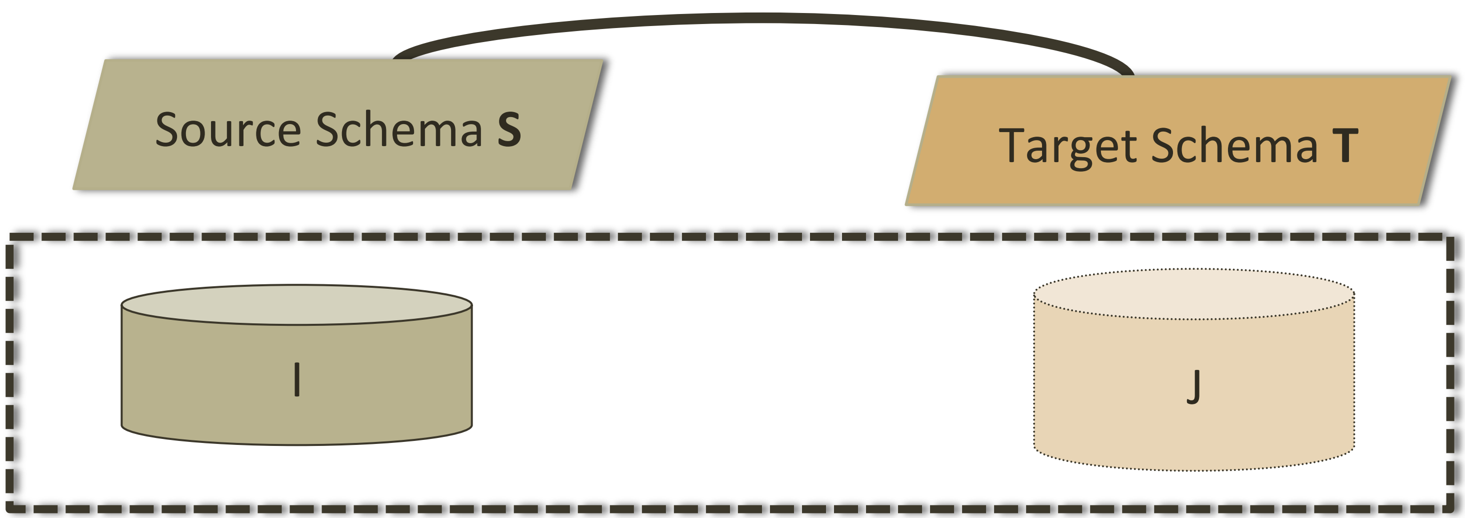
Deriving, Understanding, and Refining Schema Mappings

- Eirene: Derive, understand, and refine schema mappings via data examples
  - [Alexe, ten Cate, Kolaitis, Tan, SIGMOD 2011]
  - [Alexe, ten Cate, Kolaitis, Tan, VLDB 2011 demo]

- Muse: Understand and refine certain components of a given schema mapping via data examples
  - [Alexe, Chiticariu, Miller, Tan, ICDE 2008]
  - [Alexe, Chiticariu, Miller, Pepper, Tan, SIGMOD 2008 Demo]

Data Examples

- Recall: A *data example* is a pair (I, J) such that I is a source instance over **S** and J is a target instance over **T**.



Why Data Examples?

- Natural way to provide partial specifications of the semantics of the desired schema mapping.

Recall: "universal examples"

- User's intention: J is a *universal solution* of I w.r.t. the desired schema mapping.
  - A universal solution is a most general solution.
  - No extraneous or over-specified facts, unlike arbitrary solutions.
  - Contain just the right information needed to represent the desired outcome of migrating data.

Fitting Schema Mappings

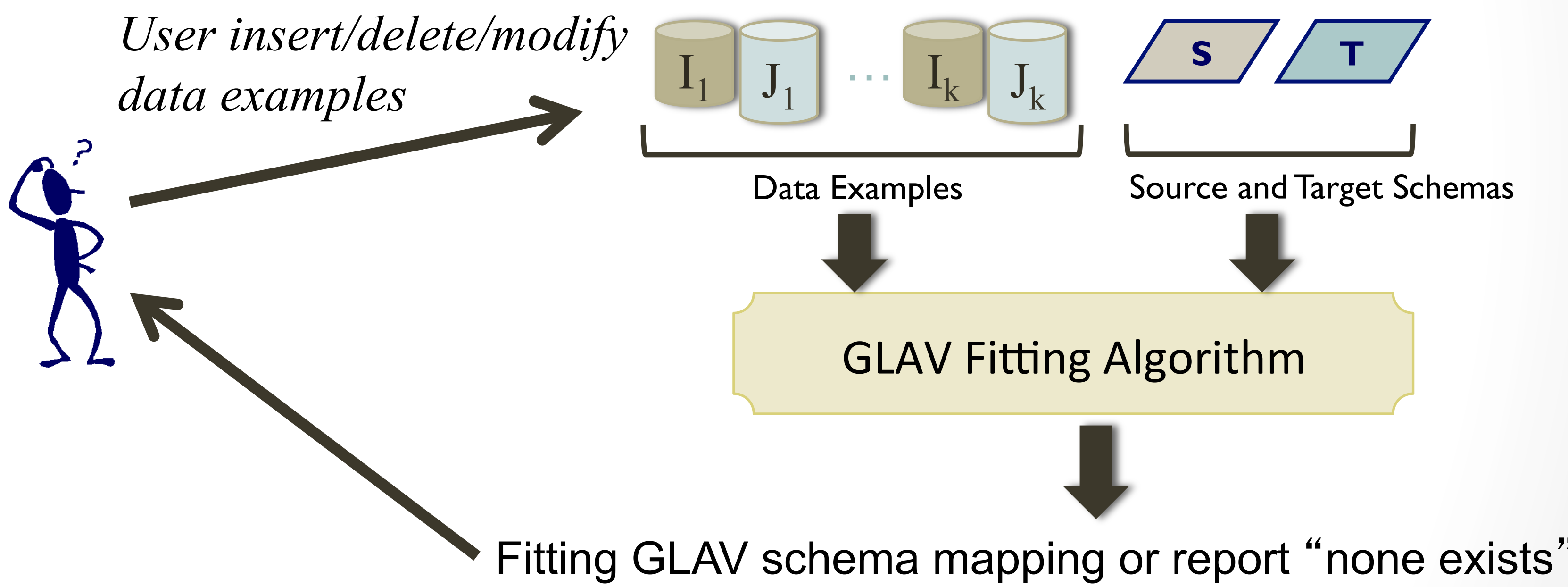
- A schema mapping *M fits a data example (I,J)* if J is a universal solution for I w.r.t. *M*.
- A schema mapping *M fits a set E of data examples* if *M* fits every data example (I,J) in *E*.

GLAV Fitting Generation Problem

Given a source schema **S**, a target schema **T**, and a finite set *E* of data examples that conform to the schemas, can we construct a GLAV schema mapping that fits *E* if possible? Otherwise, report “none exists”.

Putting the human in the loop

- Interactive design of schema mappings via data examples



An Illustration

<u>Source schema **S**</u>
Patient(pid, name, healthplan, date)
Doctor(pid, docid)

<u>Target schema **T**</u>
History(pid, plan, date, docid)
Physician(docid, name, office)

I_1:
Patient(123, Joe, Plus, Jan)
Doctor(123, Anna)

J_1:
History(123, Plus, Jan, Anna)

GLAV Fitting Algorithm

Patient(x,y,z,u) ∧ Doctor(x,v) → History(x,z,u,v)

An Illustration

<u>Source schema **S**</u>
Patient(pid, name, healthplan, date)
Doctor(pid, docid)

<u>Target schema **T**</u>
History(pid, plan, date, docid)
Physician(docid, name, office)

I_2:
Patient(123, Joe, Plus, Jan)
Doctor(123, Anna)

J_2:
History(123, Plus, Jan, N1)
Physician(N1, Anna, N2)

GLAV Fitting Algorithm

$$\text{Patient}(x,y,z,u) \wedge \text{Doctor}(x,v) \rightarrow \exists w,w' \, (\text{History}(x,z,u,w) \wedge \text{Physician}(w,v,w'))$$

"Canonical GLAV schema mapping" – based on data examples

An Illustration

<u>Source schema **S**</u>
Patient(pid, name, healthplan, date)
Doctor(pid, docid)

<u>Target schema **T**</u>
History(pid, plan, date, docid)
Physician(docid, name, office)

I_3:
Patient(123, Joe, Plus, Jan)
Doctor(123, Anna)
I_4:
Doctor(392, Bob)

J_3:
History(123, Plus, Jan, N1)
Physician(N1, Anna, N2)
J_4:
Physician(Bob, 392, N3)

GLAV Fitting Algorithm

No fitting schema mapping exists!
Intuition: The way Anna gets mapped from I_3 to J_3 contradicts the way Bob gets mapped from I_4 to J_4.

An Illustration

<u>Source schema **S**</u>
Patient(pid, name, healthplan, date)
Doctor(pid, docid)

<u>Target schema **T**</u>
History(pid, plan, date, docid)
Physician(docid, name, office)

I_5:
Patient(123, Joe, Plus, Jan)
Doctor(123, Anna)

J_5:
History(123, Plus, Jan, N1)
Physician(N1, Anna, N2)

I_6:
Doctor(392, Bob)

J_6:
Physician(N3, Bob, N4)

I_7:
Patient(653, Cathy, Basic, Feb)

J_7:
History(653, Basic, Feb, N5)

$\text{Patient}(x,y,z,u) \land \text{Doctor}(x,v) \rightarrow \exists w,w' \, (\text{History}(x,z,u,w) \land \text{Physician}(w,v,w'))$
$\text{Doctor}(x,y) \rightarrow \exists w,w' \, \text{Physician}(w,y,w')$
$\text{Patient}(x,y,z,u) \rightarrow \exists w \, \text{History}(x,z,u,w)$

GLAV Fitting Algorithm

Input: **S**, **T**, *E*

Output: A fitting GLAV schema mapping or "none exists"

1. Perform *homomorphism extension test on* every pair (I_1,J_1), (I_2,J_2) of data examples in *E*.
   If the test fails, return "none exists".

2. Construct a *fitting canonical GLAV schema mapping* *M*.
   Return *M*.

Homomorphism Extension

- A *homomorphism* h: $I_1 \rightarrow I_2$ between instances is function from adom(I_1) to adom(I_2) s.t. for every fact $P(a_1,\ldots,a_m)$ in I_1, we have that $P(h(a_1),\ldots,h(a_m))$ is a fact in I_2.

I_5:
Patient(123, Joe, Plus, Jan)
Doctor(123, Anna)

I_6:

Doctor(392, Bob)

J_5:
History(123, Plus, Jan, N1)
Physician(N1, Anna, N2)

J_6:

Physician(N3, Bob, N4)

The source homomorphism can be extended.

Homomorphism Extension

- A *homomorphism* h: $I_1 \rightarrow I_2$ between instances is function from adom(I_1) to adom(I_2) s.t. for every fact $P(a_1,\ldots,a_m)$ in I_1, we have that $P(h(a_1),\ldots,h(a_m))$ is a fact in I_2.

I_3:
Patient(123, Joe, Plus, Jan)
Doctor(123, Anna)

I_4:
Doctor(392, Bob)

J_3:
History(123, Plus, Jan, N1)
Physician(N1, Anna, N2)

J_4:
Physician(Bob, 392, N3)

The source homomorphism cannot be extended.

GLAV Fitting Algorithm: Properties

Correctness

<u>Theorem</u>: Let E be a finite set of data examples. TFAE:

1) The canonical GLAV schema mapping of E fits E.

2) There is a GLAV schema mapping that fits E.

3) For all (I, J), (I', J') $\in E$, every homomorphism h : I → I' extends to a homomorphism h' : J → J'.

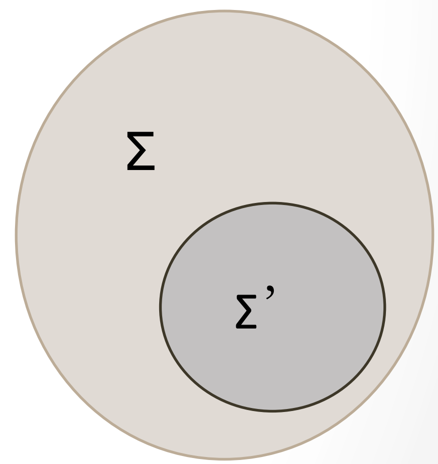
GLAV Fitting Algorithm: Properties

Most general fitting schema mapping

Theorem: Let *E* be a finite set of data examples. If there is a GLAV schema mapping that fits *E*, then the canonical GLAV schema mapping of *E* is the most general schema mapping that fits *E*.

We say that a schema mapping M is *more general* than M' if Σ' logically implies Σ.

- If for every data example (I, J) such that (I, J) satisfies Σ' we have that (I, J) also satisfies Σ.



GLAV Fitting Algorithm: Properties

Completeness for GLAV Schema Mapping Design

Theorem: For every GLAV schema mapping M, there is a finite set E_M of data examples, where M is the most general GLAV schema mapping (up to logical equivalence) that fits E_M.

GLAV Fitting Algorithm: Properties

Complexity

- Step 1 of the GLAV fitting algorithm can take exponential time.
  - Number of homomorphisms between two database instances can be exponential.
  - Every homomorphism extension must be verified in the successful case.
  - Polynomial amount of memory (for storing homomorphisms).

<u>Theorem</u>

The GLAV Fitting Generation Problem is Π_2^p-complete.

A further note

Input: **S**, **T**, *E*

Output: A fitting GLAV schema mapping or "none exists"

1. Perform *homomorphism extension test on* every pair (I_1,J_1), (I_2,J_2) of data examples in *E*.
   If the test fails, return "none exists".

2. Construct a *fitting canonical GLAV schema mapping* *M*.
   Return *M*.

Fact: For any "consistent" set of data examples, a (canonical) fitting GAV schema mapping can be computed in linear time.

Statistics of real-life mapping scenarios

	# of source relations	Avg. source arity	# of target relations	Avg. target arity	# of GLAV constraints	Avg. # of LHS atoms	Avg. # of RHS atoms
DBLP - Amalgam	7	6.5	9	6.5	10	1.4	2.2
Amalgam S1 - S2	15	6.7	27	2.0	71	1.2	2.1
GUS - BioSQL	7	6.4	6	5.5	8	1.6	1.9

# of canonical examples	Time to generate canonical examples (s)	Avg. # of nonempty source relations	Avg. # of tuples per source relation	Avg. # of nonempty target relations	Avg. # of tuples per target relation
10	4.8	1.4	1.0	2.2	1.1
15	9	1.9	1.0	10.7	1.1
7	2.3	1.6	1.1	2.1	2.3

Experimental evaluation with real-life mapping scenarios

- Canonical data examples

	Number of examples	Size of each example (# of source + target tuples)	Initial fitting test (s)	Fitting test per user change (s)
DBLP - Amalgam	10	3.8	1.6	0.2
Amalgam S1 - S2	15	13.4	3.6	0.3
GUS - BioSQL	7	6.5	1.2	0.2

Experimental evaluation with real-life mapping scenarios

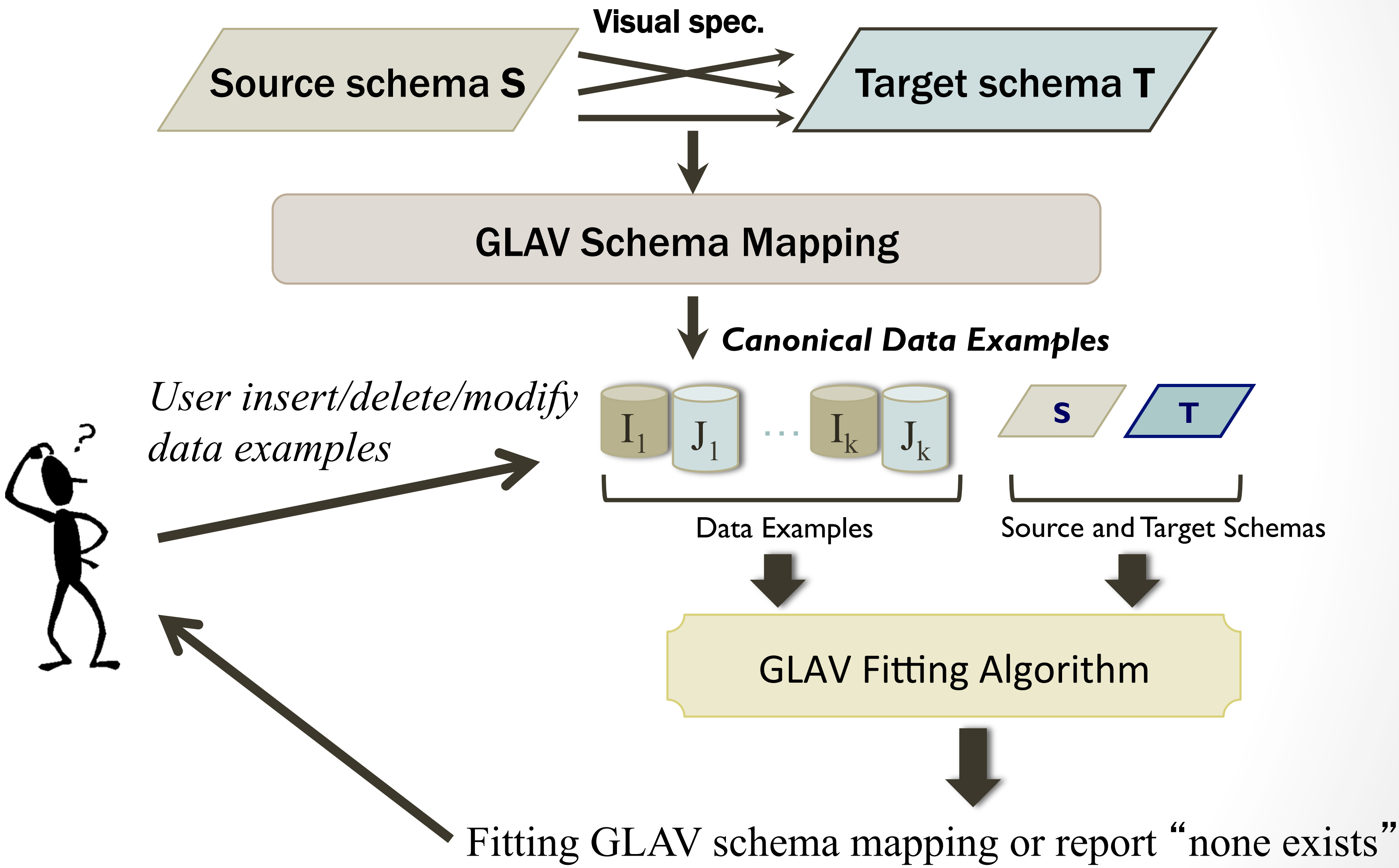
- Synthetic data examples

	Number of examples	Size of each example (# of source + target tuples)	Initial fitting test (s)	Fitting test per user change (s)
DBLP - Amalgam	10	48	17.7	1.8
Amalgam S1 - S2	10	126	222.4	23.1
GUS - BioSQL	10	39	14.2	1.5

Implementation

- Implementation over DB2 shows promising results for real schema mapping scenarios.
- Canonical Data Examples:
  - Initial execution of GLAV fitting algorithm: 1-4 secs
  - After modifications, time to refit a schema mapping, if it exists: 8% to 17% of initial fitting time.
    - Intuition: only part of the homomorphism extension test is recomputed.
- Synthetic Data Examples:
  - Initial execution of GLAV fitting algorithm: 14-222s
  - After modifications, time to refit a schema mapping, if it exists: about 10% of initial fitting time.

As part of existing Schema-Mapping Design Systems

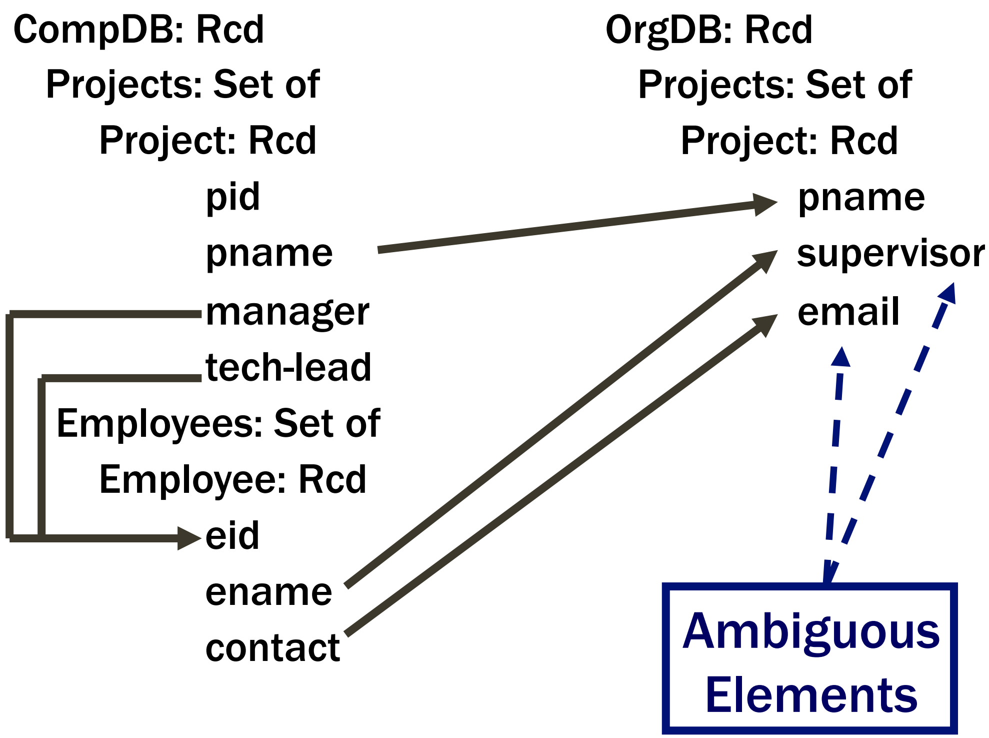


MUSE

Muse

- A mapping design wizard that uses data examples to assist designers in understanding and refining a schema mapping.
- Use of data examples was advocated in
  [Yan, Miller, Haas, Fagin SIGMOD 01]
  - Data examples are used to resolve ambiguities in visual specification, and refine mappings (SQL).
- Focus on two important components of a mapping design:
  - the specification of the desired grouping semantics for sets of data, and
  - the choice among alternative interpretations for semantically ambiguous mappings.

Ambiguous Mappings



for
    p in CompDB.Projects
    e1 in CompDB.Employees
    e2 in CompDB.Employees
satisfy
    e1.eid = p.manager
    e2.eid = p.tech-lead
exists
    p1 in OrgDB.Projects
where
    p.pname = p1.pname
    p1.supervisor =
        e1.ename **or** e2.ename
    p1.email =
        e1.contact **or** e2.contact

- This mapping is ambiguous.
- There are four alternative interpretations.

e1.ename	e1.ename	e2.ename	e2.ename
e1.contact	e2.contact	e1.contact	e2.contact

Muse-D: Disambiguating Mappings

- **Key idea**: provide a data example that illustrates the alternative interpretations in a compact way.

Projects

P1	DB	e4	e5

Employees

e4	John	john@ibm
e5	Anna	anna@ibm

Designer makes two choices

Projects

DB	John Anna	john@ibm anna@ibm

- The mapping designer makes one choice for each ambiguous element
- Each decision removes one ambiguity.
  - E.g., choosing "Anna" as the supervisor and "john@ibm" as the email.

p1.supervisor =
~~e1.ename **or**~~ e2.ename

p1.email =
e1.contact **or** ~~e2.contact~~

Obtaining Source Examples

Running queries over the real source instance.

Query:

Projects(p1,pn1,e1,e2) and

Employees(e1,en1,cn1) and

Employees(e2,en2,cn2) and

en1 != en2 and

cn1 != cn2

Non-empty result

Empty result

Real Example:

Projects

P1	**DB**	**e4**	**e5**

Employees

e4	**John**	**john@ibm**
e5	**Anna**	**anna@ibm**

Synthetic Example:

Projects

p1	**pn1**	**e1**	**e2**

Employees

e1	**en1**	**cn1**
e2	**en2**	**cn2**

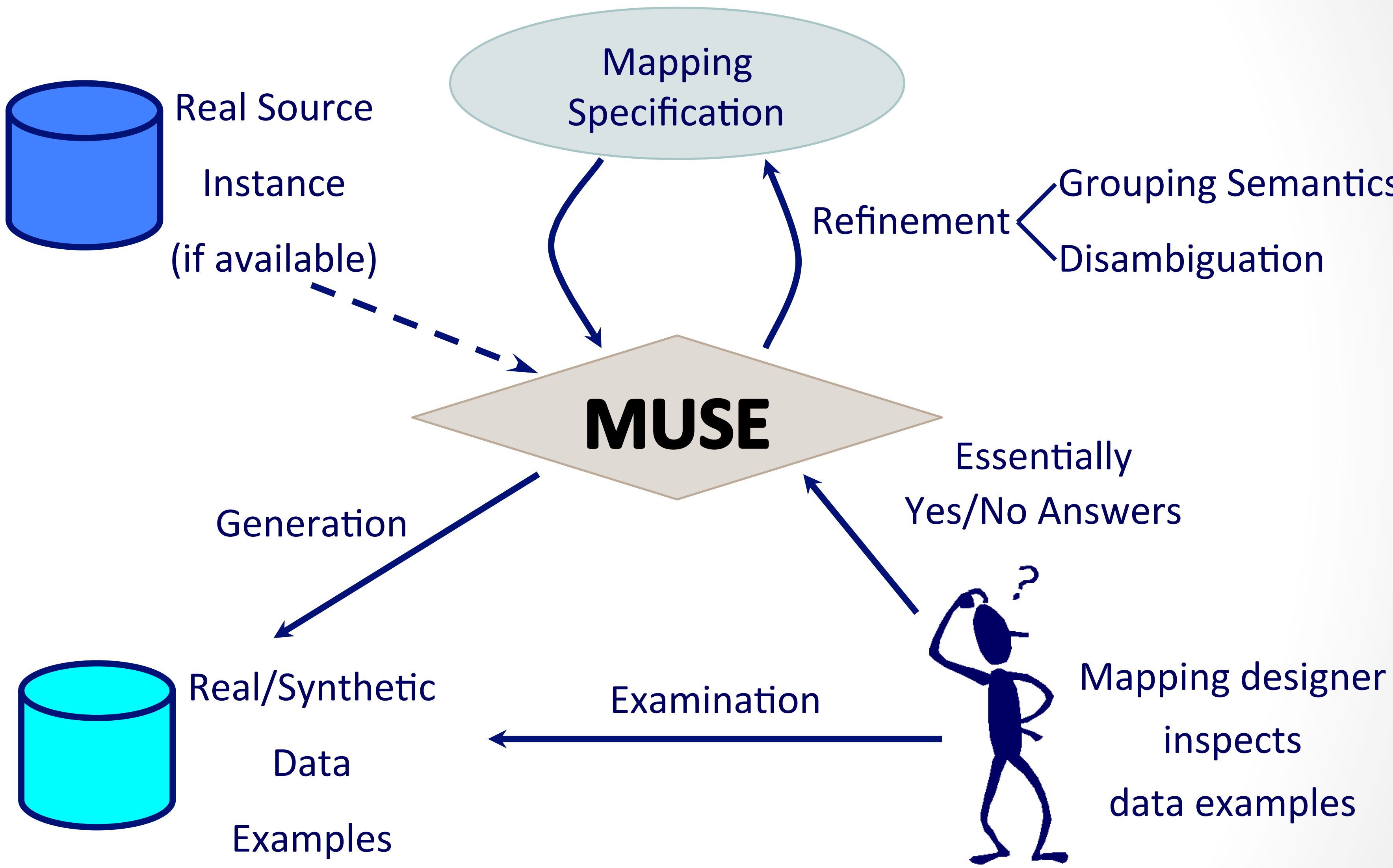
Muse-D: Properties

- For each ambiguous mapping, the designer is presented with a single compact data example.

- **Proposition (Completeness).**
  - The single data example differentiates among all the alternative interpretations of the ambiguous mapping.
  - The number of choices a mapping designer has to make is equal to the number of ambiguous elements.

- **Proposition (Small examples).** The number of tuples in the example source instance is the number of conjuncts in the for clause of the mapping.

MUSE Workflow



Schema Mappings and Data Examples

Part IV: More Approaches to Deriving Schema Mappings from Examples

EDBT'13 tutorial

Balder ten Cate, Phokion Kolaitis and Wang-Chiew Tan

Where are we?

- Two aspects of the use of data examples in schema mapping design:

  I. Using data examples to illustrate (candidate) schema mappings

  II. Deriving schema mappings from data examples

- Three approaches to deriving schema mappings from data examples:

  1. **Fitting approach (EIRENE)**: Construct a (most general) fitting schema mapping (if it exists) [Alexe - ten Cate - Kolaitis - Tan SIGMOD'11]

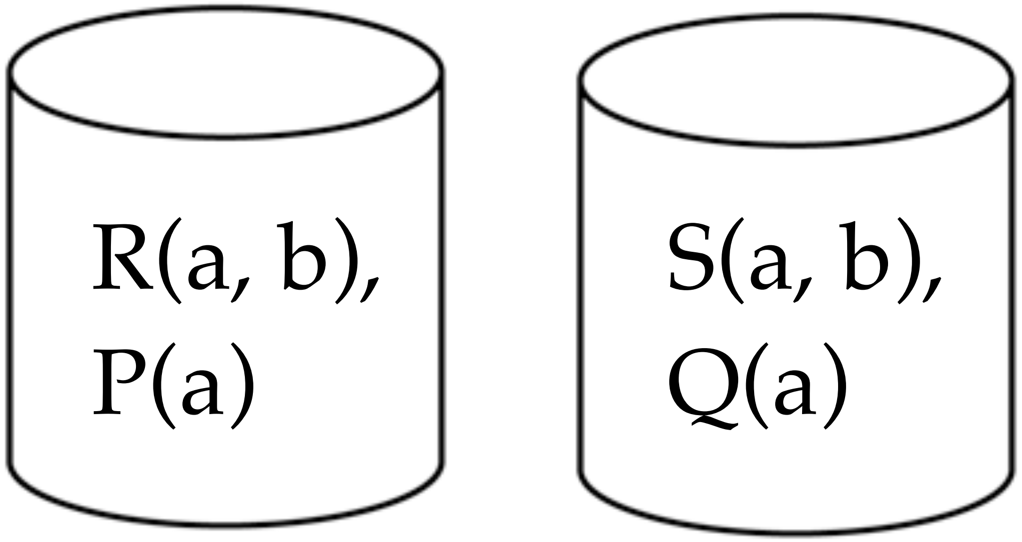
  2. **Gottlob-Senellart approach**: computing a schema mapping of "optimal cost" [Gottlob - Senellart JACM'10]

  3. **Learning Schema Mappings**: computational learning approach [ten Cate - Dalmau - Kolaitis ICDT'12]

The Fitting Approach

- We want to derive a GLAV schema mapping on the basis of a collection of (universal) data examples $(I_1, J_1), \ldots, (I_n, J_n)$.
  - **Case 1**: There is a unique fitting GLAV schema mapping
  - **Case 2**: There are multiple fitting GLAV schema mappings
  - **Case 3**: There is no fitting GLAV schema mapping

Multiple Fitting Schema Mappings



Source Target

- Schema mapping M_1:

$$\forall xy(R(x,y) \rightarrow S(x,y))$$
$$\forall x(P(x) \rightarrow Q(x))$$

- Schema mapping M_2:

$$\forall xy(R(x,y) \wedge P(x) \rightarrow S(x,y)$$
$$\forall xy(R(x,y) \wedge P(x) \rightarrow Q(x)$$

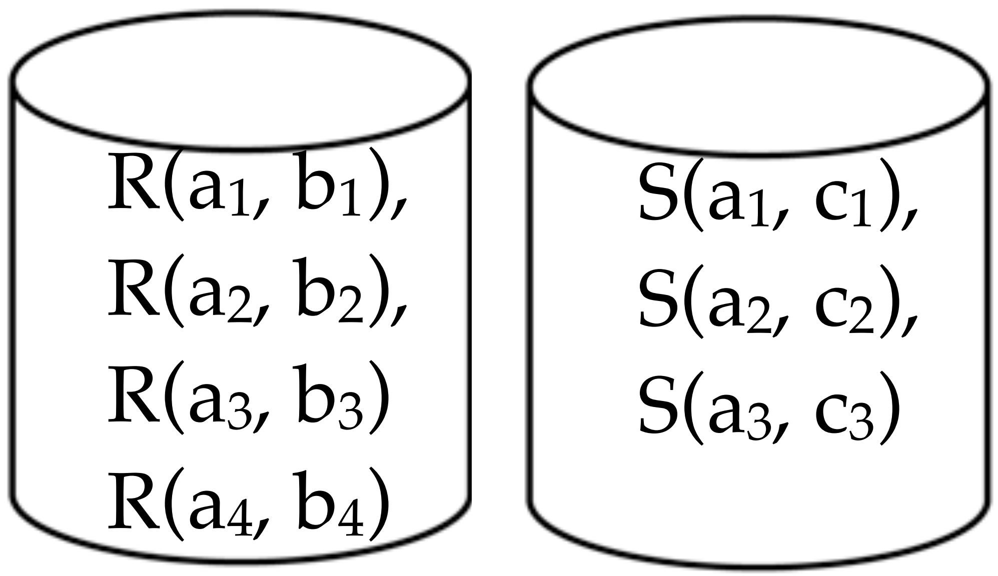
- Schema mapping M_3:

$$\forall xy(R(x,y) \rightarrow S(x,y) \wedge Q(x)$$

Most general fitting GLAV schema mapping

smallest fitting GLAV schema mapping

No Fitting Schema Mapping



Source Target

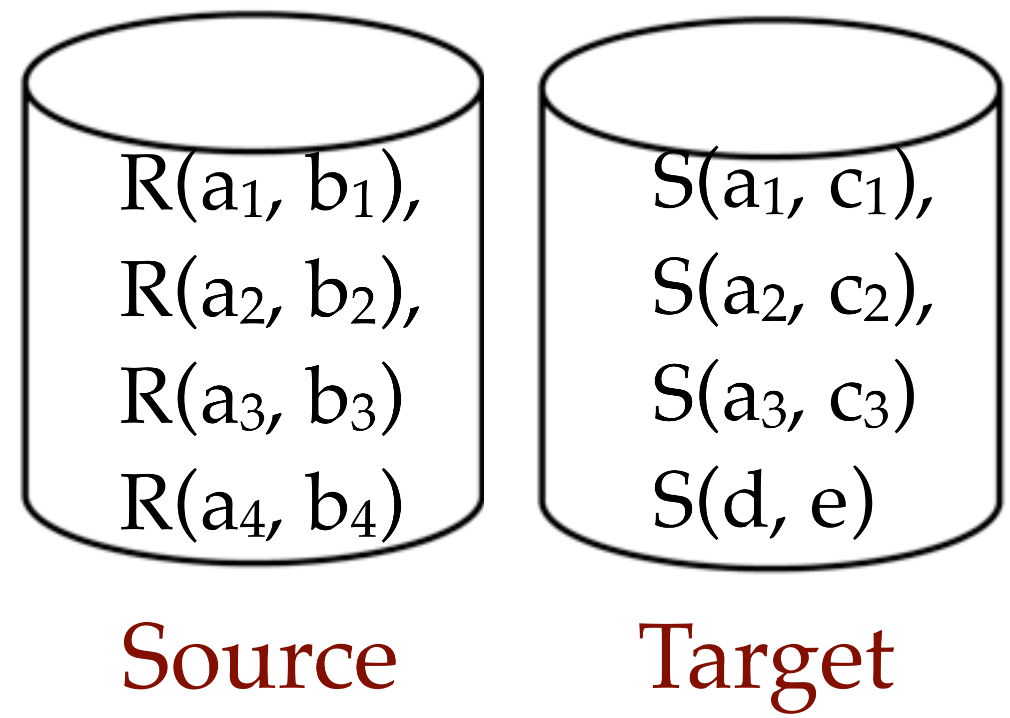
The Fitting Approach (Summary)

- **Input**: a finite collection of data examples (typically small; hand-crafted or system-generated; possibly containing labeled null values)
- **Method**: Test if a fitting GLAV schema mapping exists (homomorphism extension test, Π_2^p-complete)
  - **Yes?** Produce most general fitting GLAV schema mapping (PTIME)
  - **No?** show user where the homomorphism extension test fails, so that they can correct the examples.
- (Similarly for GAV.)

The “Gottlob-Senellart” Model

- **Input:** single data example (large; no labeled nulls; for example (DBLP,GoogleScholar))
- **Method**: find a schema mapping of “optimal cost”
  - Cost model (intuitively): takes into account size of the schema mapping and how well it fits the data example.
  - Cost model (more formally): the cost of a schema mapping is the size of the smallest “repair” that fits the given data example.
- **Two-layered approach:**
  - The basic language of GLAV schema mapping (as usual)
  - A richer language of “repaired GLAV schema mappings”

Example



- **GLAV Schema Mapping**:

  $$R(x, y) \to \exists z S(x, z)$$

- **Repaired GLAV Schema Mapping** (which fits the data example):
  - $R(x, y) \wedge x \neq a_4 \to \exists z S(x, z) \wedge \bigwedge_i (x = a_i \to z = c_i)$
  - $S(d, e)$

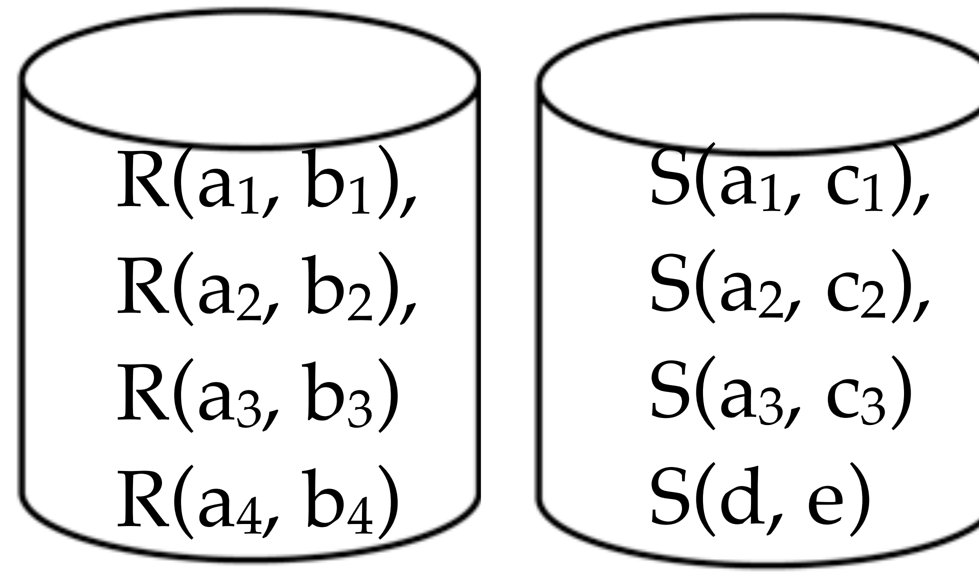
A Note on Terminology

- G&S speak of "a schema mapping M that is valid and fully explaining for (I,J)". Since (I,J) is assumed to be a ground data example, we can equivalently say that "M fits (I,J)".

Gottlob-Senellart Cost Model

- **Repair of a GLAV schema mapping** M is obtained by
  - extending left-hand sides of GLAV constraints with additional conjuncts of the form $x=c$ and $x\neq c$
  - extending right-hand side of GLAV constraints with additional conjuncts of the form $(x_1=c_1 \wedge \ldots \wedge x_n=c_n) \rightarrow y=d$
  - adding ground facts to the schema mapping
- The **size of a repaired GLAV schema mapping** is the total number of occurrences of variables and constant symbols, where ground facts $R(a_1, \ldots, a_n)$ count as having size 3n.
- The **cost of a GLAV schema mapping M** w.r.t. a data example (I,J) is the size of the smallest repair of M that fits (I,J).

Example (Revisited)



Source Target

- **GLAV Schema Mapping M**:

  $$R(x, y) \rightarrow \exists z S(x, z)$$

- **Repaired GLAV Schema Mapping M′** (which fits the data example):
  - $R(x, y) \wedge x \neq a_4 \rightarrow \exists z S(x, z) \wedge \bigwedge_i (x = a_i \rightarrow z = c_i)$
  - $S(d, e)$

$\text{Cost}_{(I,J)}(M) = \text{Size}(M') = 24$

Optimization Problem

- the problem of deriving a schema mapping from a data example becomes an optimization problem:
  - find a GLAV schema mapping M such that $cost_{(I,J)}(M)$ is mimimal.

  Such a schema mapping M is said to be “optimal” for (I,J).

Justification of the Cost Model

- Recall that GLAV schema mappings allow us to "express" the basic relation algebraic operations such as selection, projection, and join (e.g., the projection π_i is naturally "expressed" by $R(\mathbf{x}) \to S(x_i)$).

- Let γ be the (binary) relational algebra operator of

  selection, projection, union, intersection, product, or join

  and let M_γ be the schema mapping that "expresses" γ. Then, for all "sufficiently rich" instances I, we have that M_γ is optimal for (I, γ(I)).

Selected Complexity Results

- **Computing the cost of a schema mapping:**
  - Testing if $\mathrm{Cost}_{(I,J)}(M) < k$ is in Σ_3^p and Π_2^p-hard.
  - For schema mappings without ∃-quantifiers, it is in Σ_2^p and DP-hard.
- **Finding schema mappings of a given cost:**
  - Testing if there is an M with $\mathrm{Cost}(I,J)(M) < k$ is in Σ_3^p and NP-hard.
  - For schema mappings without ∃-quantifiers, it is in Σ_2^p and NP-hard.
- **Testing optimality:**
  - Testing if a given schema mapping M is optimal is in Π_4^p and DP-hard.
  - For schema mappings without ∃-quantifiers, it is in Π_3^p and DP-hard.

Pros and Cons of the GS Model

- Gottlob-Senellart Model:
  - **Pro:** always results in a schema mapping (in the worst case, M=∅)
  - **Pro:** tolerant to noise in the data example
  - **Con**: sensitive to precise definition of cost function
  - **Con**: may produce a non-fitting schema mapping even when a fitting schema mapping exists.
- See [Gottlob - Senellart JACM 2010] for more details.

Where are we?

- Two aspects of the use of data examples in schema mapping design:

  I. Using data examples to illustrate (candidate) schema mappings

  II. Deriving schema mappings from data examples

- Three approaches to deriving schema mappings from data examples:

  1. **Fitting approach**: Computing a (most general) fitting schema mapping (if it exists) [Alexe - ten Cate - Kolaitis - Tan SIGMOD'11]

  2. **Gottlob-Senellart approach**: computing a schema mapping of "optimal cost" [Gottlob - Senellart JACM'10]

  3. **Learning Schema Mappings**: computational learning approach [ten Cate - Dalmau - Kolaitis ICDT'12]

Learning Schema Mappings

- We now consider the problem of obtaining a schema mapping from data examples from the perspective of computational learning theory.
- **Our aim**: to leverage the rich body of work on learning theory in order to develop a framework for exploring the power and the limitations of the various algorithmic methods for obtaining schema mappings from data examples.
- We restrict attention to GAV schema mappings.

GAV schema mappings

- We consider a relational source schema **S** and target schema **T**.
- A GAV schema mapping M is a schema mapping specified by a finite set of GAV constraints $\forall \mathbf{x}\,(\varphi(\mathbf{x}) \rightarrow R(x_{i1},\ldots,x_{in}))$.
- We denote the set of GAV schema mappings over **S** and **T** by GAV(**S**,**T**).
- **Our main question**: under what standard models of learning are GAV schema mappings learnable using data examples?

Types of data examples

- We focus on positive and negative examples for convenience of exposition. All results also hold for universal examples.
  - In the GAV setting (unlike in the GLAV setting), positive and negative examples and universal examples are interchangeable for present purposes.
- Recall:
  - A positive example for M is a pair of instances (I,J) $\models$ M
  - A negative example for M is a pair of instances (I,J) $\not\models$ M
  - A universal example for M is a pair of instances (I,J) such that J is a universal solution for I w.r.t. M.

Computational learning theory

in time polynomial in the representation of c^g and the size of the examples returned by the oracle

exactly or approximately

- **Task**: to efficiently identify an unknown "goal concept" $c^g : X \rightarrow \{0,1\}$, for instance
  - a Boolean function ($c^g : \{0,1\}^n \rightarrow \{0,1\}$), specified by a DNF formula
  - a formal language ($c^g : \Sigma^* \rightarrow \{0,1\}$), specified by a DFA

  after asking a number of queries about it to an oracle.

"Is it the case that $c^g(x)=1$?" (membership query)

"Is it the case that $c^g \equiv c$? Give me a counterexample." (equivalence query)

"Give me a randomly generated labeled example $(x,c^g(x))$" (random example query)

Well-known models of learning

- Efficient exact learnability with membership queries and/or equivalence queries (Angluin)

  (After asking polynomially many membership/equivalence queries, the algorithm identifies the goal concept with certainty.)

  - E.g., monotone DNF formulas are efficiently exactly learnable with membership and equivalence queries. Both types of queries are needed.

- Efficient PAC (Probably-Approximately-Correct) learnability with random example queries and possibly membership queries (Valiant)

  (For all probability distributions D over the example space, when given labeled random examples drawn from D, with high probability, the algorithm produces a hypothesis that has a small expected error on random examples drawn from D.)

  - E.g., monotone DNF formulas are efficiently PAC learnable with membership queries. Membership queries are needed (assuming RP≠NP).

Exact learning vs PAC learning

- References for the (non)-learnability results for Monotone DNF: [Angluin '87; '90, Alekhnovich et al. '08].
- Relationship between exact learnability and PAC learnability [Angluin'87]:
  - Efficient exactly learnability with equivalence queries *implies* efficient PAC learnability
  - Efficient exact learnability with equivalence queries and membership queries *implies* efficient PAC learnability with membership queries
- **Caveat**: this assumes that the evaluation problem is in PTime (i.e., given a concept c and an example x, we can efficiently test if $c(x)=1$).

Our main results

Exact learning models

- GAV(**S**,**T**) is efficiently exactly learnable with membership queries and equivalence queries.
- Both types of queries are needed, unless **S** has only unary relations.

Approximate learning models

- GAV(**S**,**T**) is not efficiently PAC learnable (assuming RP $\neq$ NP), unless **S** has only of unary relations.
- GAV(**S**,**T**) *is* efficiently PAC learnable with membership queries and an oracle for NP.

Computing a fitting GAV schema mapping of near minimal length

- One cannot approximate efficiently, up to a polynomial, the shortest GAV schema mapping fitting a given set of data examples.

- All (non-)learnability continue to hold if we consider only uniquely characterizable GAV schema mappings.

Exact learnability

- **Theorem**: GAV(**S**,**T**) is efficiently exactly learnable with membership and equivalence queries.
- Proof sketch:
  - Let M^g to be the (unknown) goal GAV schema mapping.
  - Our algorithm will work by maintaining a hypothesis GAV schema mapping M^h such that $M^g \models M^h$. Initially, $M^h = \emptyset$, and after polynomially many iterations, provably $M^h \equiv M^g$.
  - **NB**: the algorithm cannot even evaluate a hypothesis M^h on an example, as the evaluation problem is coNP-hard. On the other hand, the algorithm *can* evaluate M^g on an example (membership query).

- **Definition**: For two GAV constraints C, C′, we write $C \rightarrow C'$ if the left- and right-hand side of C can be homomorphically mapped into the left- and right-hand side of C′.

  Example:

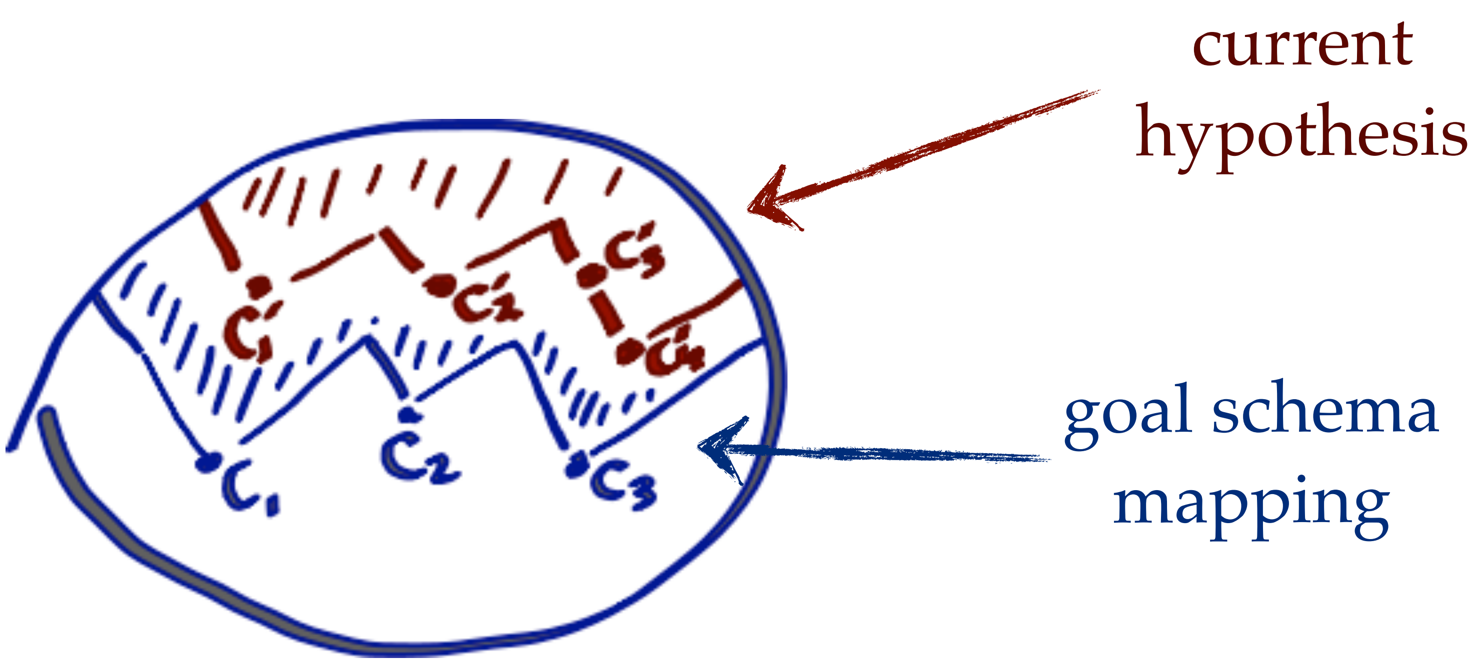
  C $\quad Rxy \wedge Ryz \rightarrow Txz$

  C′ $\quad Rxx \wedge Sx \rightarrow Txx$

- **Lemma**: Let M,M′ be sets of GAV constraints and C,C′ GAV constraints. Then

  (i) $M \models C$ if and only if $C' \rightarrow C$ for some $C' \in M$.

  (ii) $M \models M'$ if and only $\forall C' \in M' \; \exists C \in M \,.\, (C \rightarrow C')$



Idea 1: maintain an “under-approximation” of the goal schema mapping (the initial hypothesis is the empty schema mapping)

Idea 2: any counterexample to an equivalence query can be efficiently transformed (using membership queries) into new constraint C'_{i+1} that we add to the current hypothesis.

Idea 3: in polynomially many steps, we arrive at c^g.

Exact learnability (summary)

- **Theorem (stated again)**: GAV(**S**,**T**) is efficiently exactly learnable with membership and equivalence queries.

- **Theorem**: GAV(**S**,**T**) is not efficiently exactly learnable with membership queries, unless **S** contains only unary predicates.

  (Combinatorial argument: exponentially many data examples may be needed in order to identify the goal schema mapping with certainty.)

- **Theorem**: GAV(**S**,**T**) is not efficiently exactly learnable with equivalence queries, unless **S** contains only unary predicates.

  (Reduction from the analogous problem for Monotone DNF formulas [Angluin '87; '90].)

PAC learnability

- PAC (Probabilistically Approximately Correct) learning algorithm:
  - Input: a natural number n bounding the size of the goal concept, and rationals $\delta>0$ and $\varepsilon>0$.
  - Algorithm has access to an oracle that generates labeled random examples according to some probability distribution.
  - For every goal concept of size at most n, and for every probability distribution D, the algorithm, when given labeled random examples drawn from D, produces with high probability ($1-\delta$), a hypothesis that has a small expected error (ε) on random examples drawn from D.
  - The algorithm terminates in time polynomial in $1/\delta$, $1/\varepsilon$, n, and the maximal size of a labeled example returned by the oracle.

PAC learnability

- **Theorem**: GAV(**S**,**T**) is not efficiently PAC learnable, unless **S** contains only unary relations.

  The proof is based on a reduction from non-PAC learnability of monotone DNF formulas [Alekhnovich et al. '08]

- **Theorem**: GAV(**S**,**T**) is efficiently PAC learnable with membership queries and an oracle for NP.

  Obtained as a consequence of the exact learnability result (the need for an NP oracle reflects the hardness of checking whether a candidate schema mapping fits a given data example)

Approximating the Smallest Fitting Schema Mapping

- Call a set of labeled examples consistent if a fitting GAV schema mapping exists.
- Recall: For any consistent set of labeled examples, a canonical fitting GAV schema mapping can be computed in linear time.
- The canonical fitting GAV schema mapping has the same order of size as the input examples. Much shorter fitting GAV schema mappings may exist.
- Can we do better? Can we compute a fitting GAV schema mapping whose size is close to minimal?

Approximating the Smallest Fitting Schema Mapping

- **Theorem**: there is no polynomial time algorithm that, given a consistent set of data examples, produces a fitting GAV schema mapping of size less than n^k, for fixed k, where n is the size of the smallest fitting GAV schema mapping (assuming RP≠NP)

- Obtained as a corollary of our non-efficient PAC learnability result (in fact we obtain a slightly stronger non-approximability result.)
- The same result holds when the input is a single universal example.

Conclusion on Learning Schema Mappings

- We studied the problem of obtaining a schema mapping from data examples from the lens of computational learning theory.
- We obtained both positive and negative results.
  - GAV schema mappings are efficiently exactly learnable, but only if both membership and equivalence queries are allowed.
  - GAV schema mappings are not PAC learnable, but they are PAC learnable with membership queries and access to an NP-oracle.
- Open questions:
  - are GAV schema mappings efficiently PAC learnable with membership queries (and without an NP-oracle)?
  - What about LAV schema mappings, and GLAV schema mappings?

Deriving Schema Mappings from Data Examples

- Further open question
  - Richer schema mapping languages (including, e.g., target constraints, data value transformation, ...)
  - Suitable definitions of "approximate fitting" for data examples, for which no fitting schema mapping exists.

Final Words

- Data example are useful in schema mapping design, understanding, refinement.
- Two main thrusts:
  - Illustrating/characterizing a (candidate) schema via data examples
  - Deriving schema mappings from examples
- The research we presented draws from different areas, such as databases, constraint satisfaction, logic, and computational learning.
- Schema mapping design can be a difficult task, and data examples constitute a helpful tool.