Schema Mappings
and
Data Examples

An Interplay of Syntax and Semantics

Phokion G. Kolaitis
UC Santa Cruz & IBM Research – Almaden

Logic and Databases

- Extensive interaction between logic and databases during the past 40 years.
- Logic provides both a unifying framework and a set of tools for formalizing and studying data management tasks.
- The interaction between logic and databases is a prime example of
  - Logic **in** Computer Science

    but also
  - Logic **from** Computer Science

The Relational Data Model

Introduced by E.F. Codd, 1969-1971

- Relational Database:
  Collection D = (R_1, …, R_m) of finite relations (tables)

- Such a relational database D can be identified with the finite relational structure **A**[D] = (adom(A), R_1, …, $R_{\mathbf{m}}$), where adom(A) is the active domain of D, i.e.,
  the set of all values occurring in the relations of D.

Two Main Uses of Logic in Databases

- Logic as a formalism for defining database query languages
  - Codd proposed using First-Order Logic as a database query language, under the name Relational Calculus.
    - First-Order Logic (and its equivalent reformulation as Relational Algebra) are at the core of SQL
  - Datalog = Existential Inductive Definability

    (a.k.a. Positive First-Order Logic + Recursion)

- Logic as a specification language for expressing database dependencies, i.e., semantic restrictions (integrity constraints) that the data of interest must obey.
  - Keys and Functional Dependencies, Inclusion Dependencies.

A More Recent Challenge: Data Interoperability

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

- Applications need to access and process all these data.

- Growing market of enterprise data interoperability tools:
  - Multibillion dollar market; 17% annual rate of growth
  - 15 major vendors in Gartner's Magic Quadrant.

A Third Use of Logic in Databases

In the past decade, logic has also been used is also used as a formalism to specify and study critical data interoperability tasks, such as

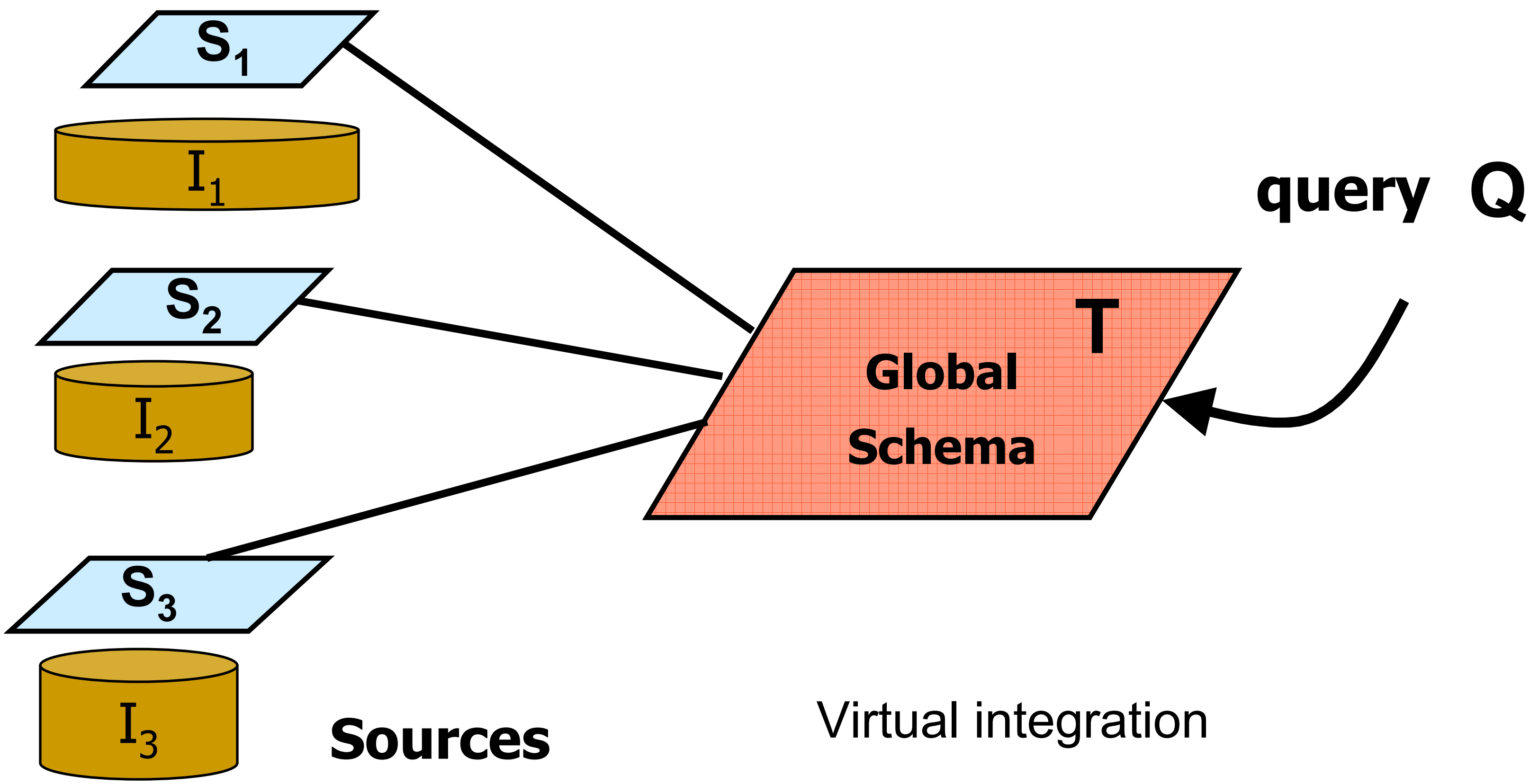
- Data Integration (aka Data Federation)

and

- Data Exchange (aka Data Translation)

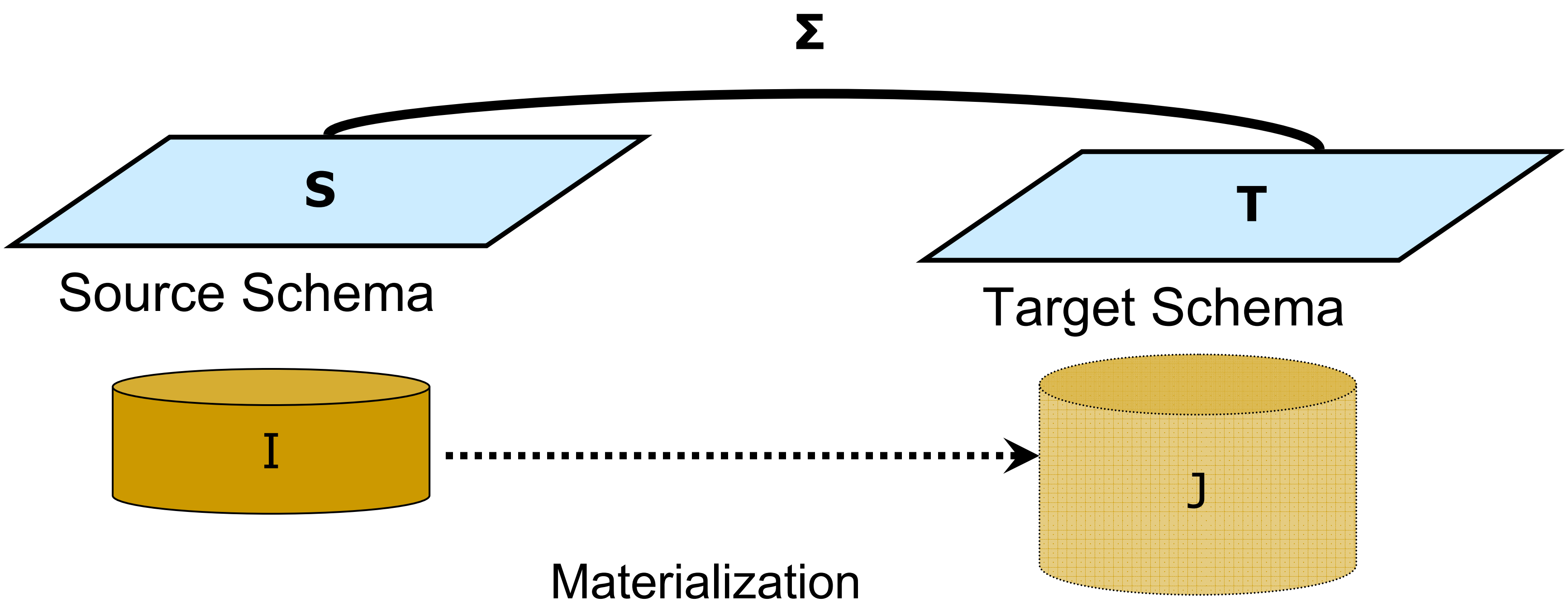
Data Integration

Query heterogeneous data in different sources via a virtual global schema



Data Exchange

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



Challenges in Data Interoperability

Fact:

- Data interoperability tasks require expertise, effort, and time.
- **Key challenge:** Specify the relationship between schemas.

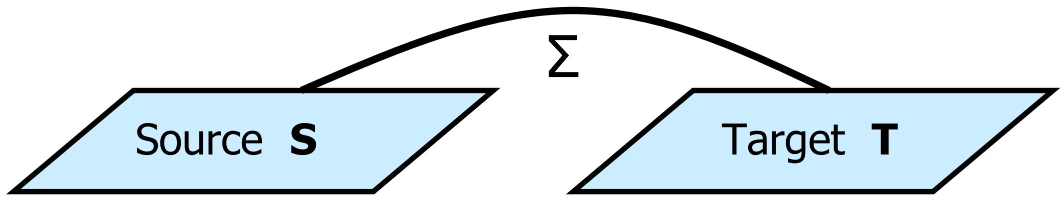
Earlier approach:

- Experts generate complex transformations that specify the relationship as programs or as SQL/XSLT scripts.
- Costly process, little automation.

More recent approach: Use **Schema Mappings**

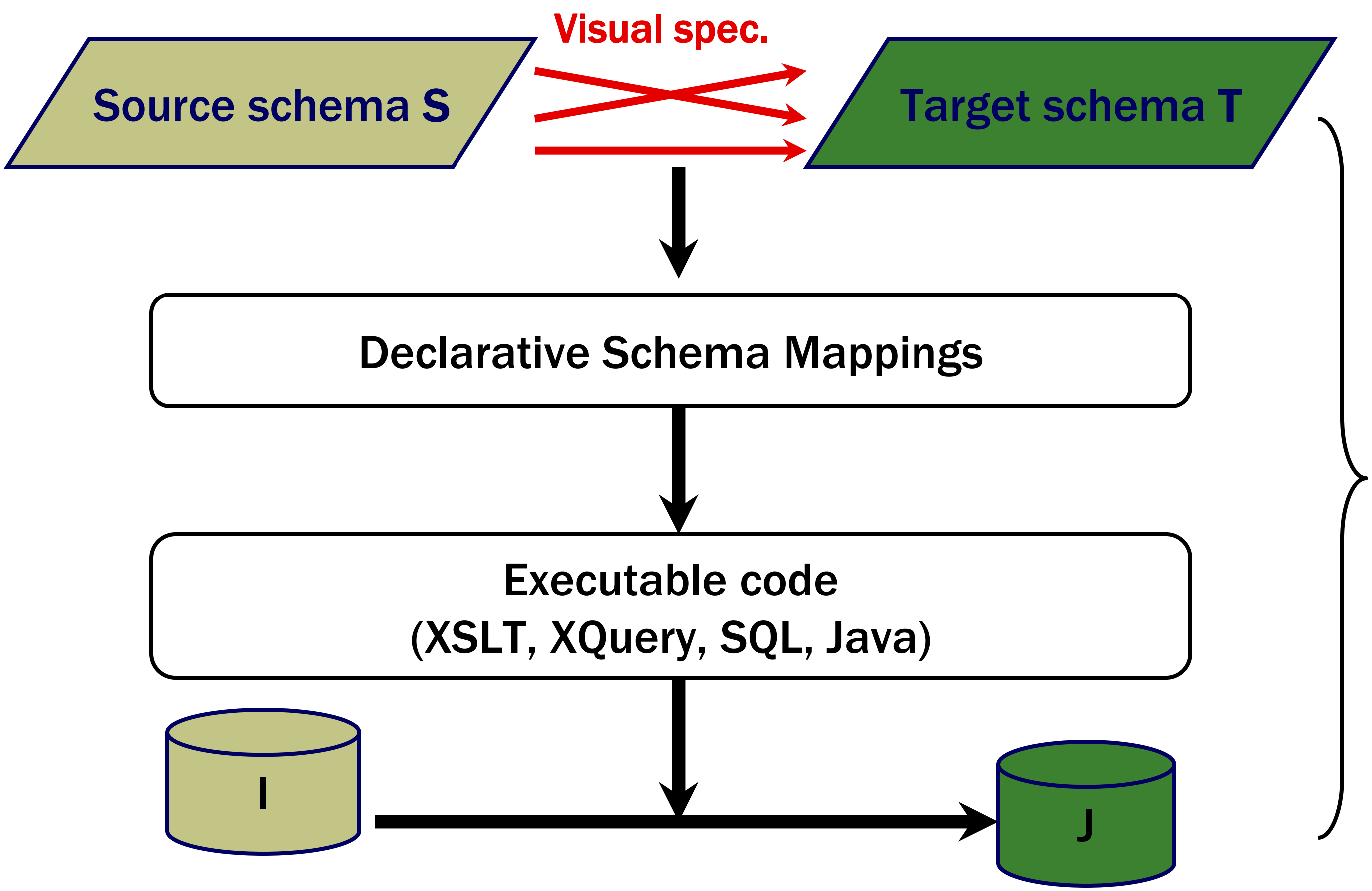
- Higher level of abstraction that separates the design of the relationship between schemas from its implementation.
- Schema mappings can be compiled into SQL/XSLT scripts automatically.

Schema Mappings



- **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 Systems: State-of-the-Art



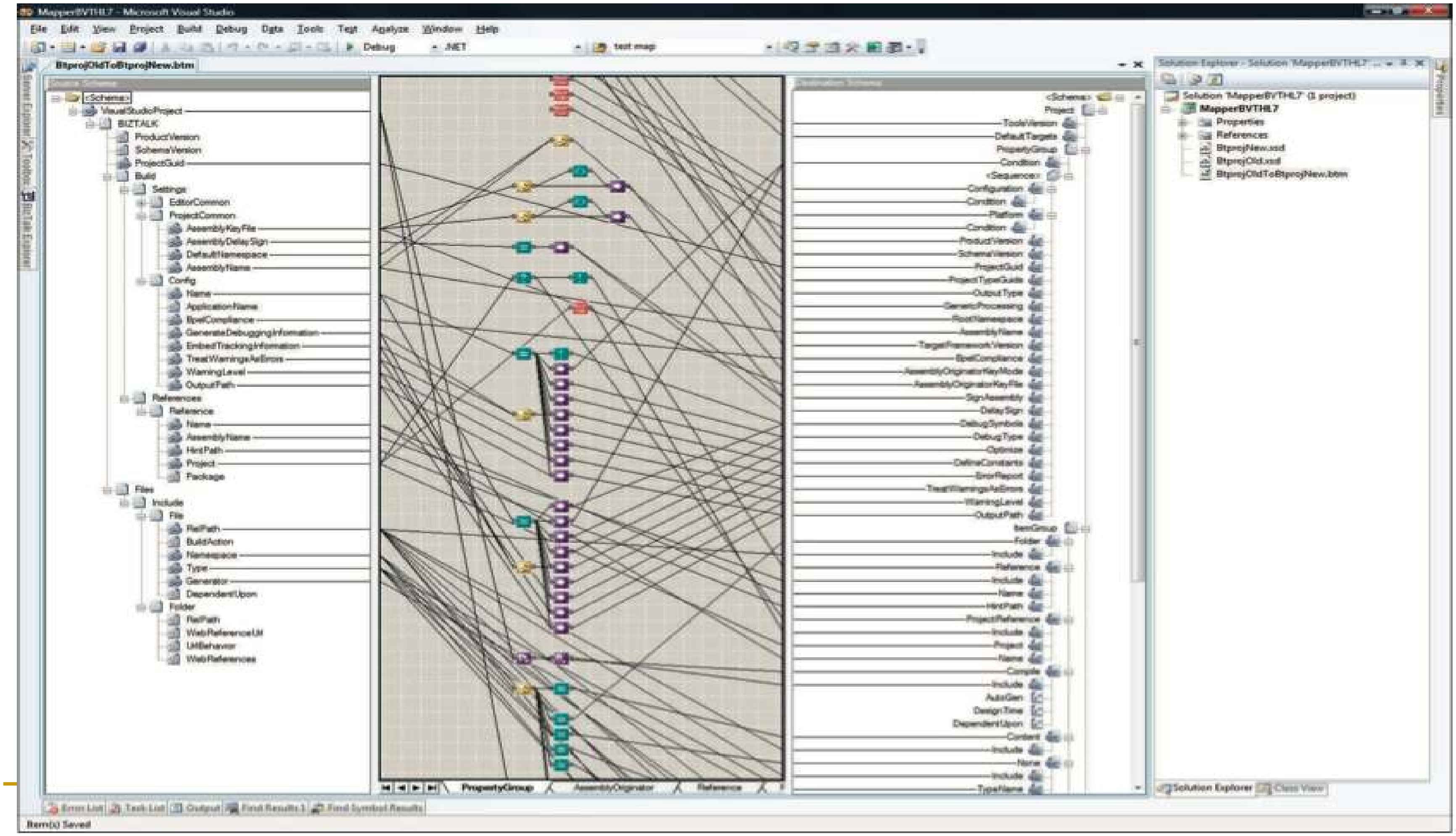
Generic architecture of schema-mapping systems
e.g.,
IBM Clio, HePToX
Altova MapForce
Stylus Studio
MS Biztalk Mapper

Schema Mappings

However, schema mappings can be **complex** …

Visual Specification

- Screenshot from the Bernstein and Haas 2008 CACM article "*Information Integration in the Enterprise*".



Schema Mappings (one of many 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 visual specification) 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".

Schema Mappings and Data Examples

Research Goals:

- Develop a framework for the systematic investigation of data examples for schema mappings.
- Understand both the **capabilities** and **limitations** of data examples in capturing, deriving, and designing schema mappings.

Collaborators and References

Bogdan Alexe, Balder ten Cate, Victor Dalmau, Wang-Chiew Tan

- Characterizing Schema Mappings via Data Examples
  ten Cate, Alexe, K …, Tan - ACM TODS 2011
  (earlier version in PODS 2010)
- Database Constraints and Homomorphism Dualities
  ten Cate, K …, Tan - CP 2010
- Designing and Refining Schema Mappings via Data Examples
  Alexe, ten Cate, K …, Tan - SIGMOD 2011
- EIRENE: Interactive Design and Refinement of Schema Mappings via Data Examples
  Alexe, ten Cate, K …, Tan - VLDB 2011 (demo track)
- Learning Schema Mappings
  ten Cate, Dalmau, K … - ICDT 2012

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 + …)
  - $\forall x,y,z(E(x,z) \wedge F(z,y) \rightarrow \exists\, w\ (R(x,y) \wedge T(x,y,z,w)))$

Schema-Mapping Specification Languages

Fact: All preceding tasks can be specified using
source-to-target tuple-generating dependencies (**s-t tgds**):

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

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

Examples:

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

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

Note: Tuple-generating dependencies (no distinction between source and target) are defined analogously.

Tuple-Generating Dependencies

They are not new:

- Extensively studied in the 1970s and the 1980s in the context of database integrity constraints (Beeri, Fagin, Vardi, ..)

  "A Survey of Database Dependencies"

  by R. Fagin and M.Y. Vardi – 1987

- "A Formal System for Euclid's Elements"
  by J. Avigad, E. Dean, J. Mumma

  The Review of Symbolic Logic – 2009

  Claim:

  All theorems in Euclid's Elements can be expressed by tuple-generating dependencies!

Tuple-Generating Dependencies

They surface in unexpected places:

- "Relational Hidden Variables and Non-Locality"
  by S. Abramsky – Studia Logica 2013

Study of foundations of quantum mechanics in a relational framework.

Fact: Many properties of quantum systems can be expressed as tuple-generating dependencies:

- No-signalling; λ-independence; Outcome independence; Parameter Independence; Locality

Example: No-signalling for 2-dimensional relational models

- $\forall x,y,z,s,t,u,v \; (R(x,y,s,t) \wedge R(x,z,u,v) \rightarrow \exists w \; R(x,z,s,w))$

"Whether an outcome s is possible for a given measurement x is independent of the other measurements."

Source-to-Target Tuple-Generating Dependencies

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

  $\forall \mathbf{x}\, (\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y}))$, where

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

  They are also known as

  GLAV (global-and-local-as-view) constraints.

- They generalize **LAV (local-as-view)** constraints:

  $\forall \mathbf{x}\, (P(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y}))$, where P is a source relation.

- They generalize **GAV (global-as-view)** constraints:

  $\forall \mathbf{x}\, (\varphi(\mathbf{x}) \rightarrow R(\mathbf{x}))$, where R is a target relation.

LAV and GAV Constraints

Examples of LAV (local-as-view) constraints:

- Copy and projection
- Decomposition: $\forall x\ \forall y\ \forall z\ (P(x,y,z) \rightarrow R(x,y) \wedge T(y,z))$
- $\forall x\ \forall y\ (E(x,y) \rightarrow \exists\ z\ (H(x,z) \wedge H(z,y)))$

Examples of GAV (global-as-view) constraints:

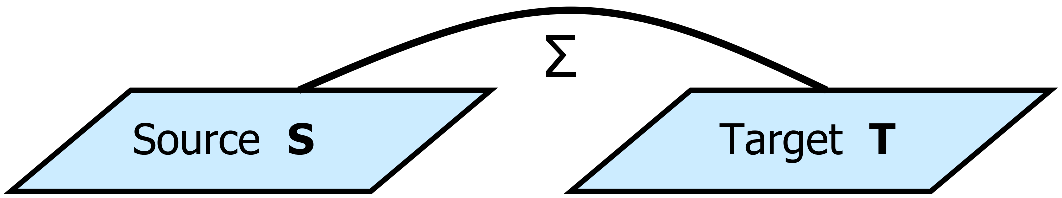
- Copy and projection
- Join: $\forall x\ \forall y\ \forall z\ (E(x,y) \wedge E(y,z) \rightarrow F(x,z))$

Note:

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

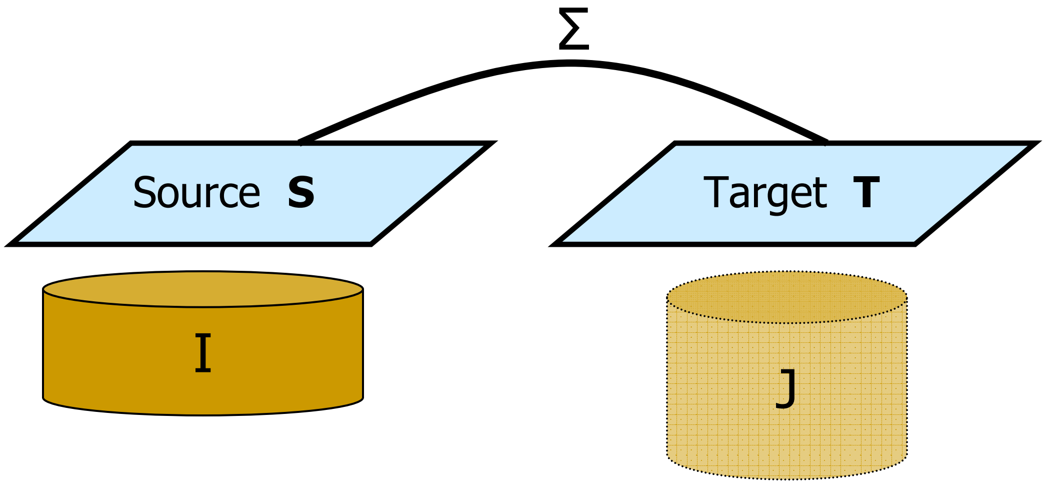
is a GLAV constraint that is neither a LAV nor a GAV constraint

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 Mapping** defined in a similar way.

Semantics of Schema Mappings

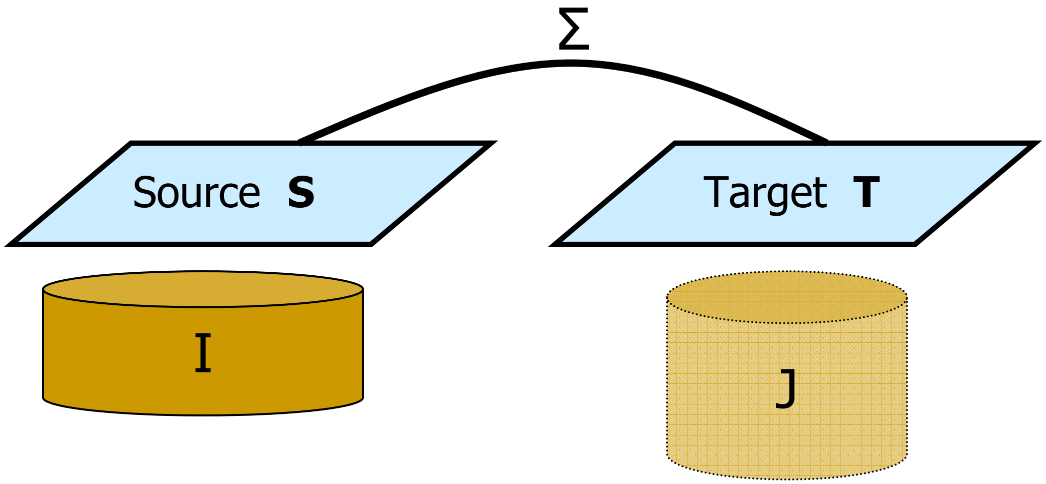


M = (**S**, **T**, Σ) a GLAV schema mapping.

- 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 $\mathbf{M} = (\{E\}, \{F\}, \Sigma)$, 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

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

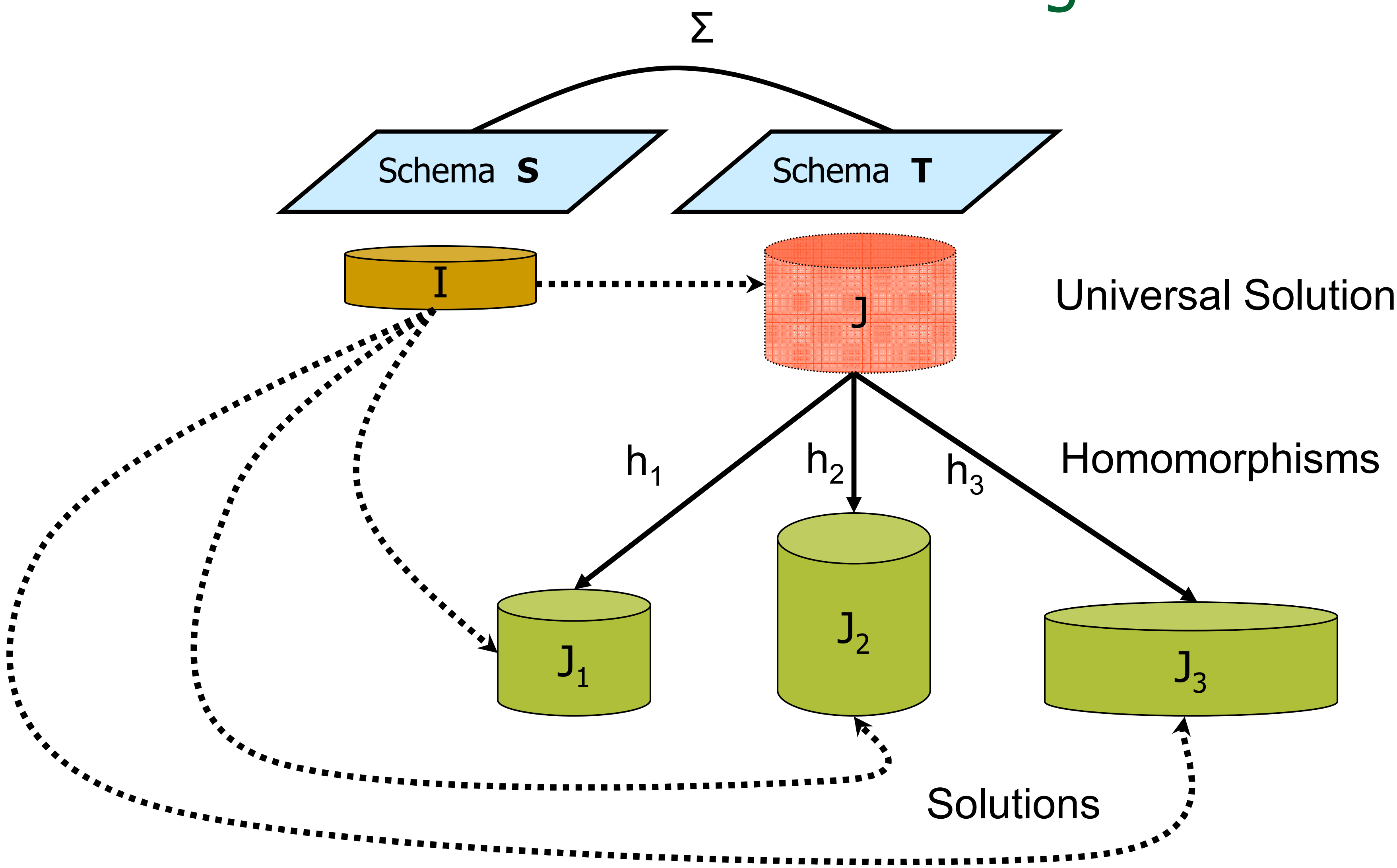
Definition: **M** = (**S**, **T**, Σ) schema mapping, I source instance.

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 $chase_{\mathbf{M}}(I)$ 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 $chase_{\mathbf{M}}(I)$ 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

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

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

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

$\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ 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: $(I^+_1, \text{chase}_{\mathbf{M}}(I^+_1))$, $(I^+_2, \text{chase}_{\mathbf{M}}(I^+_2))$, …

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

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?

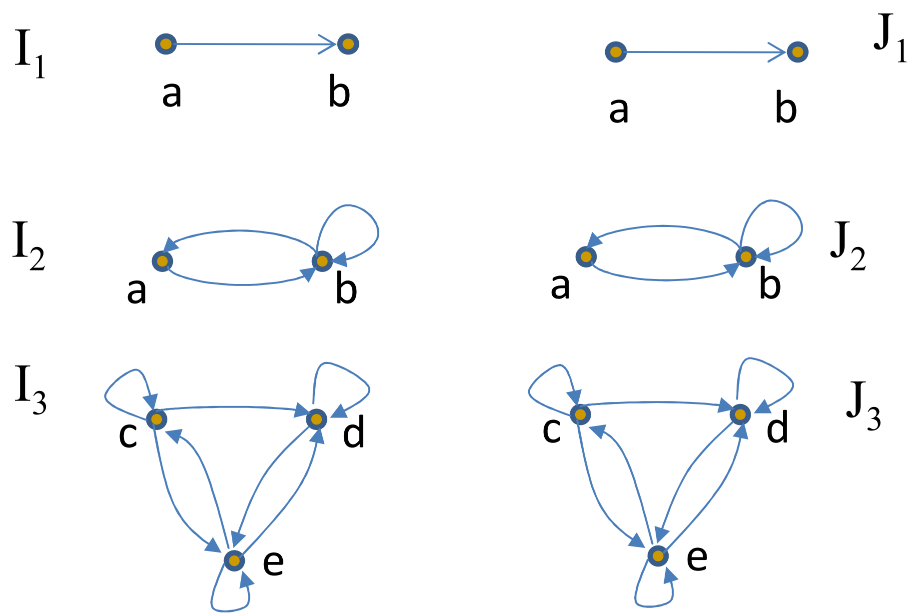
Unique Characterizations Warm-Up

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 $J = \text{chase}_{\mathbf{M}}(\{ R(c_1,\ldots,c_m) \})$, 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

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

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

    $\mathbf{A} \rightarrow \mathbf{D}$ if and only if $(\mathbf{F} \nrightarrow \mathbf{A})$.

    In symbols, $\rightarrow\mathbf{D} = \mathbf{F}\nrightarrow$
  - 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

    $$\mathbf{G} \rightarrow \mathbf{K_1} \text{ if and only if } \mathbf{K_2} \nrightarrow \mathbf{G}.$$
  - **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**

    $$\mathbf{H} \rightarrow \mathbf{T_k} \text{ if and only if } \mathbf{P_{k+1}} \nrightarrow \mathbf{H}.$$

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 $\boldsymbol{F}$ and $\boldsymbol{D}$ be two sets of structures. We say that ($\boldsymbol{F}$, $\boldsymbol{D}$) is a **duality pair** if for every structure **A**, TFAE
  - There is a structure **D** in $\boldsymbol{D}$ such that $\mathbf{A} \rightarrow \mathbf{D}$.
  - For every structure **F** in $\boldsymbol{F}$, 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 $\boldsymbol{F}$ is an **obstruction set** for $\boldsymbol{D}$.

Homomorphism Dualities

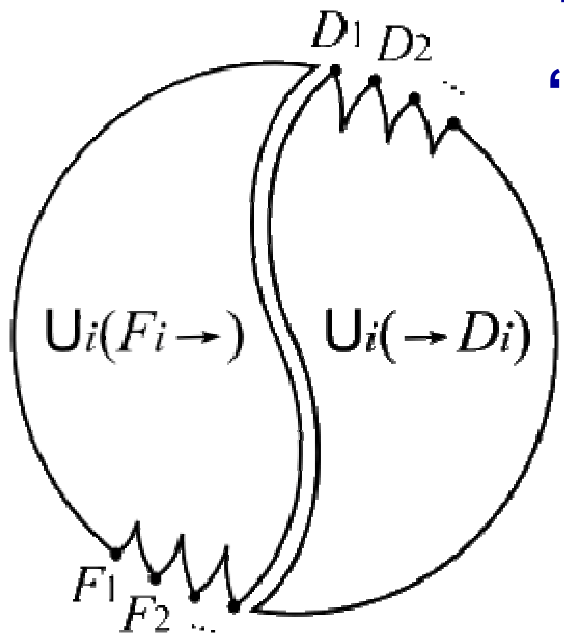
Duality Pair ($\boldsymbol{F}$,$\boldsymbol{D}$),where

$\boldsymbol{F} = \{F_1, F_2, \ldots\}$

$\boldsymbol{D} = \{D_1, D_2, \ldots\}$

The Yin

"Dreams": $\bigcup_i (\to D_i)$



The Yang

"Fears": $\bigcup_i (F_i \to)$

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

$$G \rightarrow D \text{ if and only if } F \not\rightarrow G.$$

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 **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 ***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

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

  is uniquely characterizable via universal examples.

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

  The canonical structures of these constraints are
  - $A_\sigma = (\{x,y,x\} \{E(x,y), E(y,z), E(z,x)\}, x, z)$
  - $A_\tau = (\{x,y\}, \{E(x,y), E(y,x)\}, x, x)$

  - Both are c-acyclic; hence $\{A_\sigma, A_\tau\}$ 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.

From Semantics to Syntax: Deriving Schema Mappings from Data Examples

- **The Fitting Problem for a Class C of Schema Mappings:**
  Given a finite set of data examples, is there a schema mapping in **C** for which they are universal?

- **Learnability of Schema Mappings:**
  Can we learn a **goal** schema mapping from data examples in some learning theory model?
  (e.g., Angluin's model of
  exact learning with membership queries).

Complexity & Algorithms for the Fitting Problem

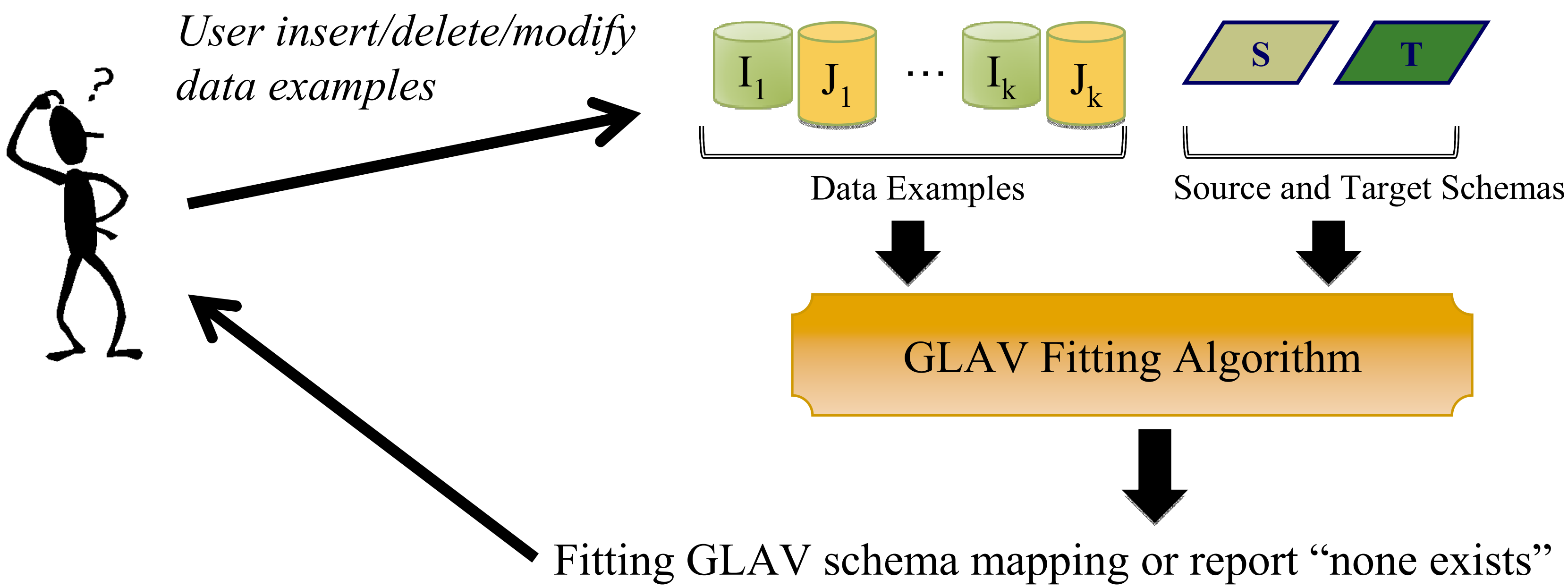
Theorem:

- The fitting problem for GAV mappings is DP-complete.
- The fitting problem for GLAV mappings is Π_2^p -complete.
- There is an algorithm, based on a homomorphism extension test, that, given a finite set of data examples,
  - Tests for the existence of a fitting mapping.
  - If there is a fitting schema mapping, then the algorithm produces the most general GAV fitting mapping or

    the most general GLAV fitting mapping, where most general means that it is implied by every other fitting mapping.

EIRENE: A System for Deriving Schema Mappings Interactively

- Interactive design of schema mappings from data examples via the fitting algorithms for GLAV and GAV mappings



Learning Schema Mappings

- Angluin's model of exact learning with membership queries is very natural in this setting.

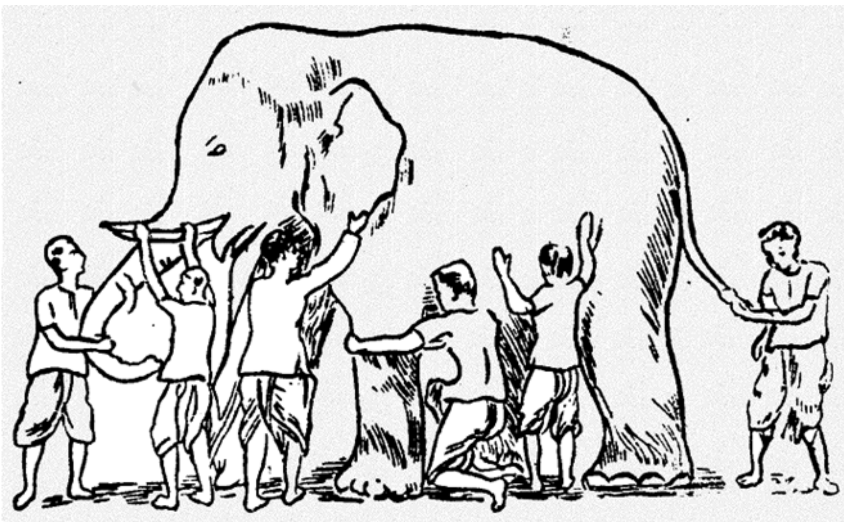
- **Schema-Mapping-Reverse-Engineering Problem:**
  We have a "black box" (object code) for performing data exchange, i.e., object code for producing, given a source instance I, a universal solution J for I. Can we use it to recover the underlying schema mapping?

Learning GAV Mappings

Theorem: Let **S** be a source schema, **T** a target schema, and let GAV(**S**, **T**) be the of all GAV mappings **M** = (**S**, **T**, Σ).

- GAV(**S**, **T**) is efficiently exactly learnable with equivalence and membership queries.
- GAV(**S**, **T**) is **not** efficiently exactly learnable with only equivalence queries or only membeship queries, unless the source schema **S** consists of unary relation symbols only.

Data Interoperability: The Elephant and the Six Blind Men



- Data interoperability remains a major challenge:

  "Information integration is a beast." (L. Haas – 2007)

- GLAV schema mappings capture some, but far from all, aspects of data interoperability.

- Much work remains to be done.

- However, mathematical theory and computational practice can inform each other.

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.