Schema Mappings and Data Examples

An Interplay between Syntax and Semantics

Phokion G. Kolaitis

UC Santa Cruz & IBM Research – Almaden

Logic and Databases

- Logic provides both a unifying framework and a set of tools for formalizing and studying data management tasks.
- The interaction between logic and databases started with the introduction of the **relational data model** by E.F. Codd in 1969.
- It continues today across a wide spectrum of topics in database management.
- This talk is about the role of logic in **data interoperability**.

The Data Interoperability Challenge

- Data may reside
  - at several different sites
  - in several different formats (relational, XML, ...).
- Applications need to access and process all these data.
- Growing market of enterprise data interoperability tools:
  - About $4B in 2012; growing at about 9% per year.
- Data interoperability is thought to consume about 40% of the budget of enterprise IT shops

  (Bernstein and Haas, CACM 2008)

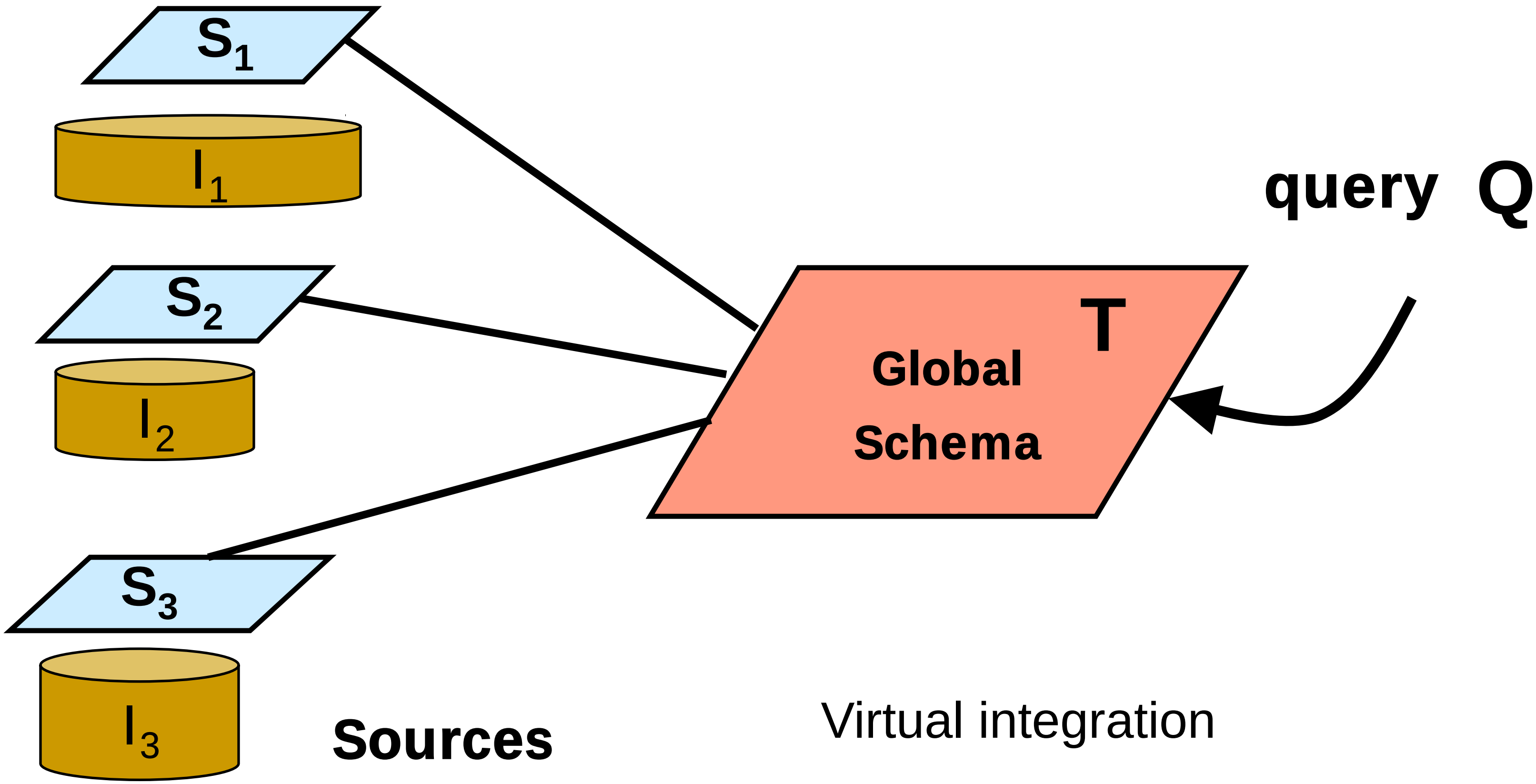
Theoretical Aspects of Data Interoperability

The research community has studied two different, but closely related, facets of data interoperability:

- **Data Integration** (aka **Data Federation**)
- **Data Exchange** (aka **Data Translation**)

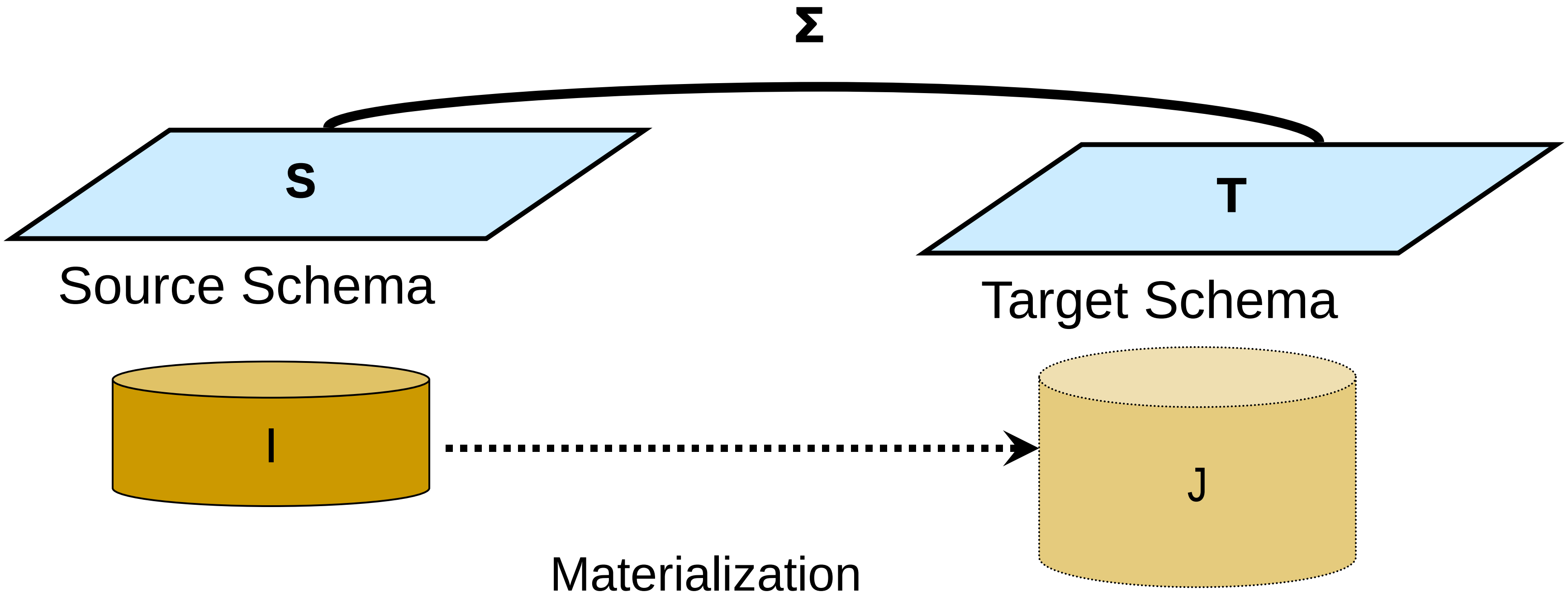
Data Integration

Query heterogeneous data in different sources via a virtual global schema



Data Exchange

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



Challenges in Data Interoperability

Fact:

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

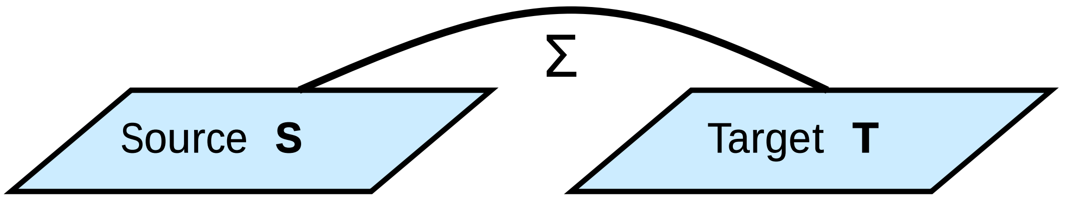
Earlier approach:

- 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**

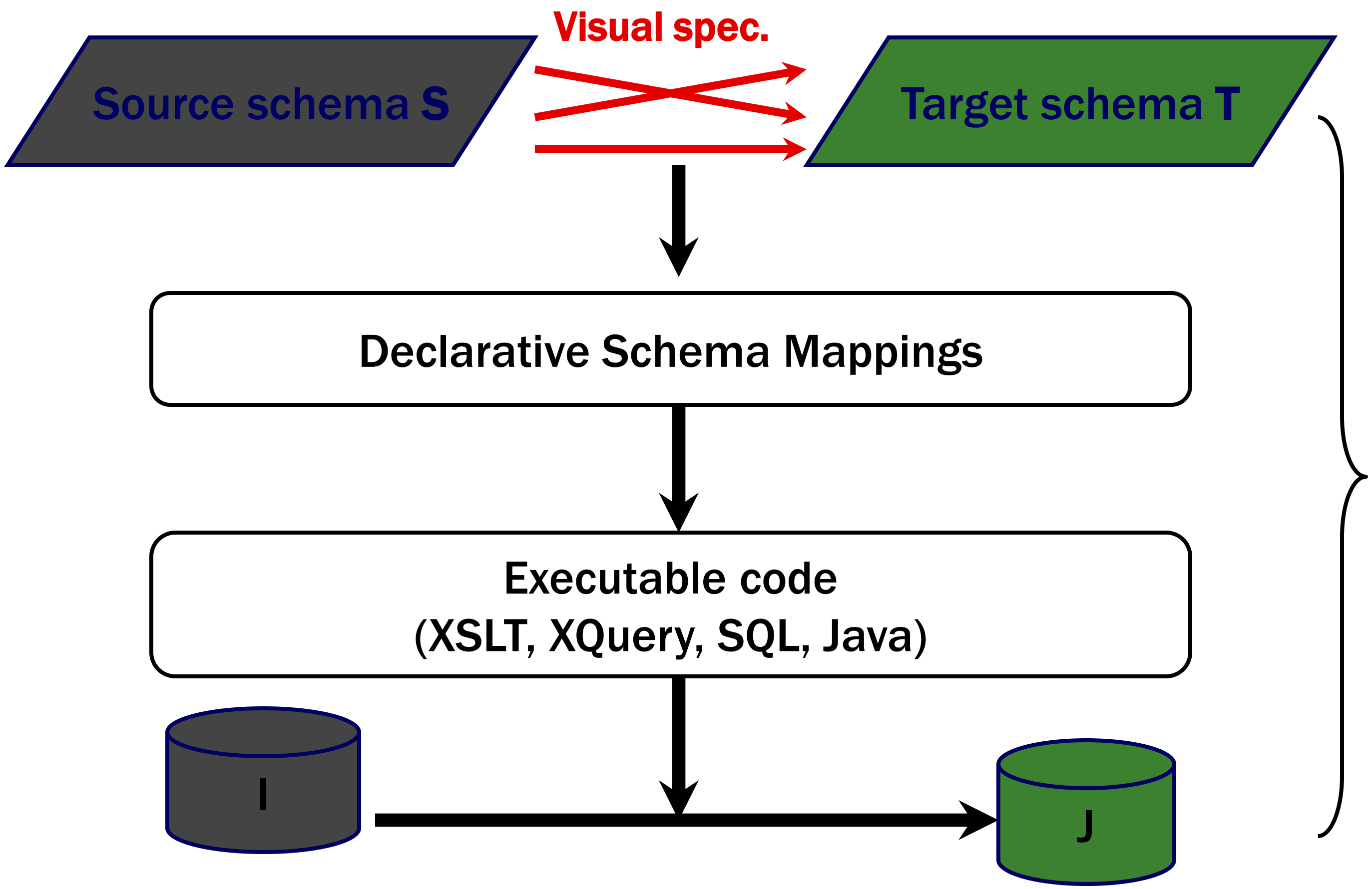
- Higher level of abstraction that separates the design of the relationship between schemas from its implementation.
- Schema mappings can be compiled into SQL/XSLT scripts automatically.

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



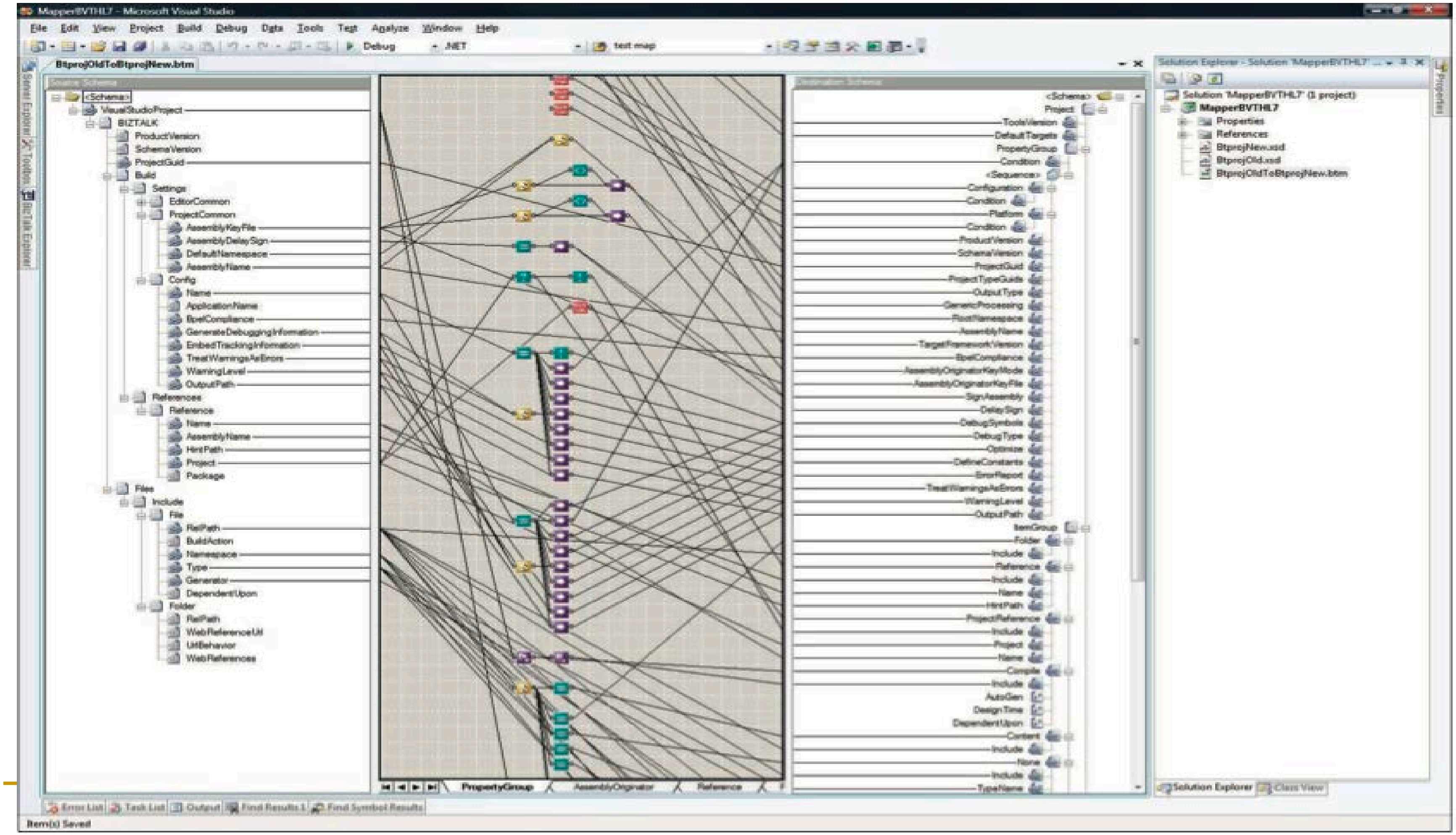
Generic architecture of schema-mapping systems
e.g.,
IBM Clio, HePToX
Altova MapForce
Stylus Studio
MS Biztalk Mapper

Schema Mappings

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

Visual Specification

- Screenshot from 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 the interaction between schema mappings and data examples.
- 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
  Alexe, ten Cate, K ..., Tan - ACM TODS 2011 (earlier 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 ... - ACM TODS 2013 (earlier in 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 + …)
  - $\forall x,y,z(E(x,z) \wedge F(z,y) \rightarrow \exists\, w\ (R(x,y) \wedge T(x,y,z,w)))$

Source-to-Target Tuple-Generating Dependencies

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

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

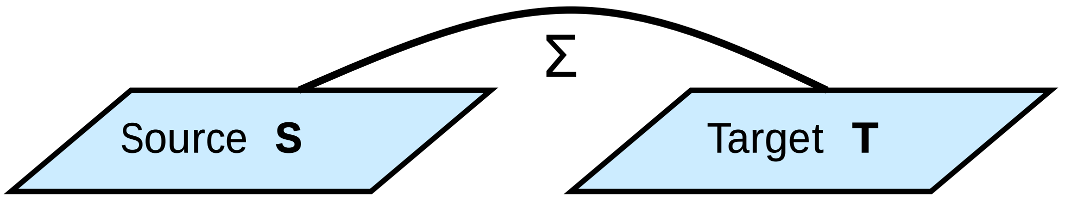
- Copy and projection
- Join: $\forall x\ \forall y\ \forall z\ (E(x,y) \wedge E(y,z) \rightarrow F(x,z))$

Note:

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

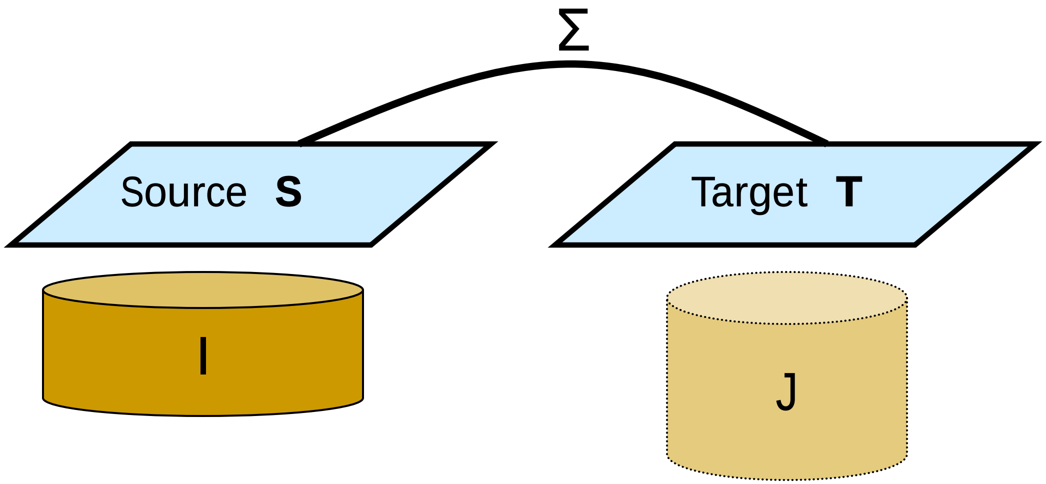
is a GLAV constraint that is neither a LAV nor a GAV constraint

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.

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) $\models$ Σ
  - In this case, we say that J is a solution for I w.r.t. **M.**

Schema Mappings and Data Examples

- **M** = (**S**, **T**, Σ) GLAV schema mapping
  - This is a finite syntactic object.
- Sem(**M**) = { (I,J): (I,J) is a positive data example for **M** }
  - Sem(**M**) is a semantic object that characterizes **M**; however, Sem(**M**) is is an infinite set of data examples.

Question:

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

Types of Data Examples

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

- 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.

GLAV Mappings and Universal Solutions

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

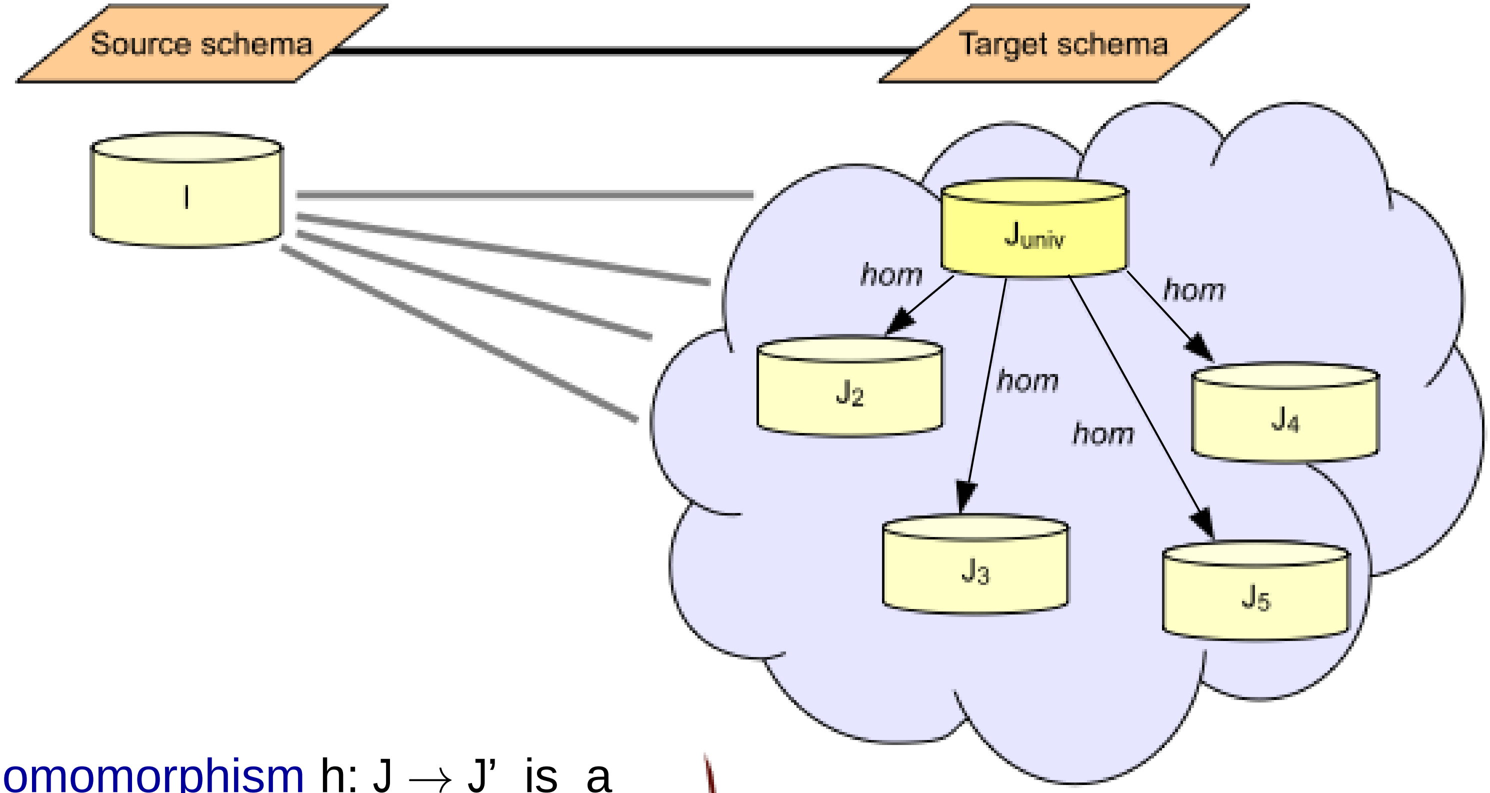
- Intuitively, universal solutions are the "most general" solutions.
- They have become the preferred semantics of data exchange.

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

- Every source instance I has a **universal solution** J **w.r.t. M**, i.e., a solution J for I such that if J' is another solution for I, then there is a homomorphism h: J $\rightarrow$ J' that is constant on adom(I)

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

Universal Solutions in Data Exchange



Defn: A homomorphism h: J → J' is a function sending every constant (non-null) value to itself, and preserving facts $P(a_1...a_n) \in J \Rightarrow P(h(a_1)...h(a_n)) \in J'$

Example

- 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,X), F(X,2) } (universal)
  - J_2 = { F(1,2), F(2,2) } (**not** universal)
  - J_3 = { F(1,X), F(X,2), F(Y,Z) } (universal)
  - J_4 = { F(1,X), F(X,2), F(Y,Y) } (**not** universal)

    (where X, Y, Z are labeled null values)
  - …

From Syntax to Semantics: Characterizing Schema Mappings via Data Examples

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

Question:

Can **M** be "uniquely 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**?

Notions of Unique Characterizability

Definition: **M** = (**S**, **T**, Σ) a GLAV schema mapping, $\mathcal{C}$ 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. $\mathcal{C}$ if for every finite set Σ' $\subseteq$ $\mathcal{C}$ such that **P** and **N** are sets of positive and negative examples for **M'** = (**S**, **T**, Σ'), we have that Σ $\equiv$ Σ'.

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

Relationships between Unique Characterizability Notions

- **Facts:**
  - 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 $\mathcal{C}$ be a class of GLAV constraints.

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

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

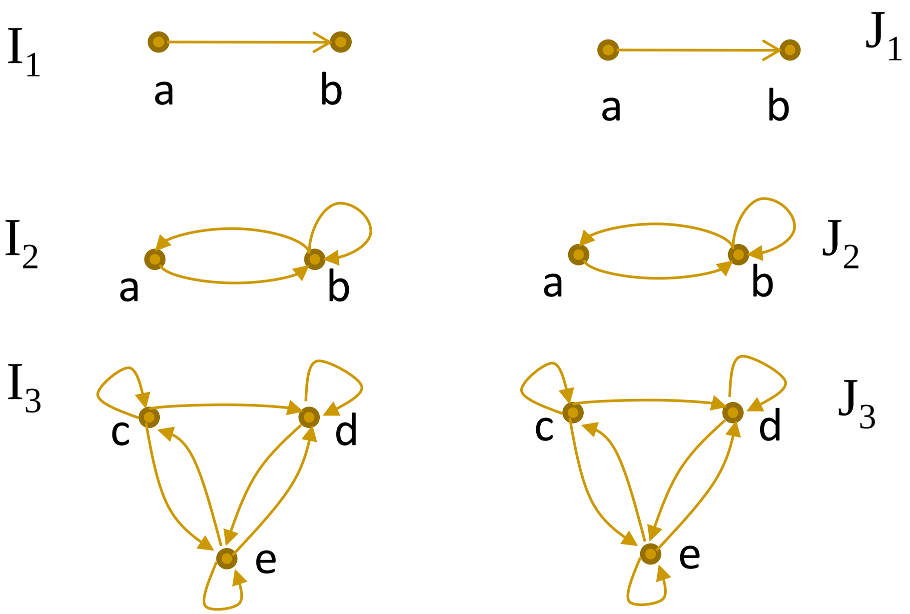
- The set **U** = { (I_1, J_1) } with I_1 = { E(a,b }, J_1 = { F(a,b) } uniquely characterizes **M** w.r.t. the class of all LAV constraints.

- 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 $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ 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.

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.

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:** $\mathbf{G} \rightarrow \mathbf{K_2}$ 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**

    $\mathbf{A} \rightarrow \mathbf{D}$ if and only if $(\mathbf{F} \not\rightarrow \mathbf{A})$.

    In symbols, $\rightarrow\mathbf{D} = \mathbf{F}\not\rightarrow$
  - 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, $\rightarrow \mathbf{K}_2 = \bigcap_{i\geq 0} (\mathbf{C}_{2i+1} \not\rightarrow)$.

- **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 **A** $\rightarrow$ **D**.
  - For every structure **F** in ***F***, we have **F** $\not\rightarrow$ **A**.

  In symbols, $\bigcup_{\mathbf{D} \in \boldsymbol{D}} (\rightarrow \mathbf{D}) = \bigcap_{\mathbf{F} \in \boldsymbol{F}} (\mathbf{F} \not\rightarrow)$.

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

Homomorphism Dualities

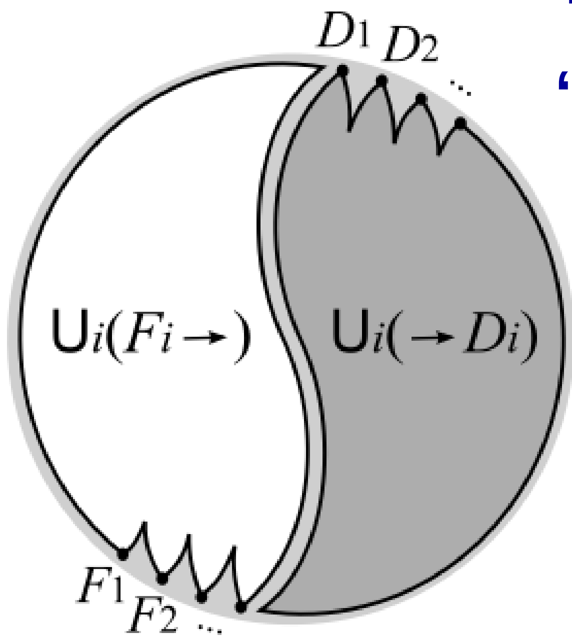
Duality Pair (***F***,***D***),where

F = {**F**$_1$,**F**$_2$,…}

D = {**D**$_1$,**D**$_2$,…}

The Yin

"Dreams": $\bigcup_i$ ($\rightarrow$D$_i$)



The Yang

"Fears": $\bigcup_i$ (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 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 $F(\mathbf{M},R)$ 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

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

  - 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

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

  - 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 of structures.

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?

Complexity of Unique Characterizations of GAV Mappings

Theorem:

- 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.

- This problem is in LOGSPACE if **M** is in normal form, i.e., ***F*** (**M**,R) consists of pairwise incomparable cores.

  - **Algorithm:** A certain acyclicity test.

Note:

- Extends results of Foniok, Nešetřil, and Tardif – 2008.
- Every GAV mapping can be transformed to a logically equivalent one in normal form.

Applications

- If **M** is a GAV mapping specified by a tgd in which all variables in the LHS are exported to the RHS, then **M** is uniquely characterizable.
  - Copy tgd: $\forall x \forall y\ (E(x,y) \rightarrow F(x,y))$

- 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.

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

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

  is uniquely characterizable.

From Syntax to Semantics

Summary:

- Necessary and sufficient condition in terms of homomorphism dualities for unique characterizations of GAV mappings
- Complexity of Decision Problem:
  - NP-complete for arbitrary GAV mappings
  - In LOGSPACE for GAV mappings in normal form.

Open Problem:

- Unique characterizations of GLAV schema mappings?
- Is it a decidable problem?

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.

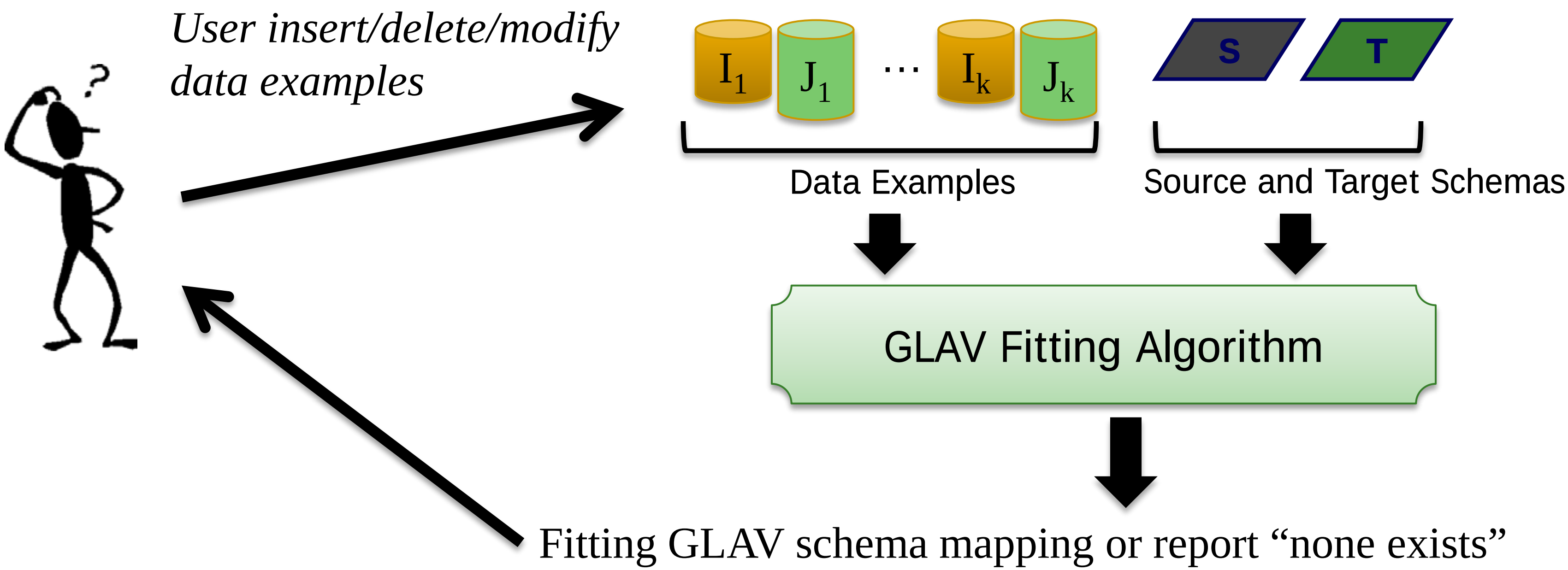
- The fitting problem for GLAV mappings is Π_2^{p}-complete.

- 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.

Concluding Remarks

Summary: Rich interplay between syntax and semantics for schema mappings and data examples:

- Unique characterizability
- Fitting problem
- Learning schema mappings from data examples.

Ongoing Work and Next Steps:

- Criterion for unique characterizability of GLAV mappings.
- Unique characterizability, fitting, and learning of schema mappings in richer languages:
  - GLAV mappings + target constraints
  - Disjunctive tuple-generating dependencies
- Schema-mapping derivation as an optimization problem in a cost model developed by Gottlob and Senellart.