A Tutorial
on
Schema Mappings & Data Exchange

DEIS ‘10

Phokion G. Kolaitis

University of California Santa Cruz
&
IBM Research - Almaden

Outline of the Tutorial

- Schema Mappings as a framework for formalizing and studying data interoperability tasks.

- Data Exchange and Solutions in Data Exchange
  - Universal Solutions and the Core.

- Query Answering in Data Exchange.

- Managing schema mappings via operators:
  - The composition operator
  - The inverse operator and its variants

Acknowledgments

- Much of the work presented has been carried out in collaboration with
  - Ron Fagin, IBM Almaden
  - Renee J. Miller, University of Toronto
  - Lucian Popa, IBM Almaden
  - Wang-Chiew Tan, UC Santa Cruz.

  Papers in ICDT 2003, PODS 2003-2010, TCS, ACM TODS.

- The work has been motivated from the Clio Project at IBM Almaden aiming to develop a working system for schema-mapping generation and data exchange.

The Information Integration 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 information integration tools:
  - About $1.5B per year; 17% annual rate of growth.
  - Information integration consumes 40% of the budget of enterprise information technology shops.

Gartner's Magic Quadrant Report on Information Integration Products

Challengers	*Leaders*
Microsoft Oracle	Informatica IBM (Cognos, Ascential) SAP – Business Objects
Sybase Syncort ETI Pitney Boss Software Open Text	SAS Pervasive Software iWay Software Sun Microsystems Tibco Software
Niche Players	*Visionaries*

Ability to execute

Completeness of vision

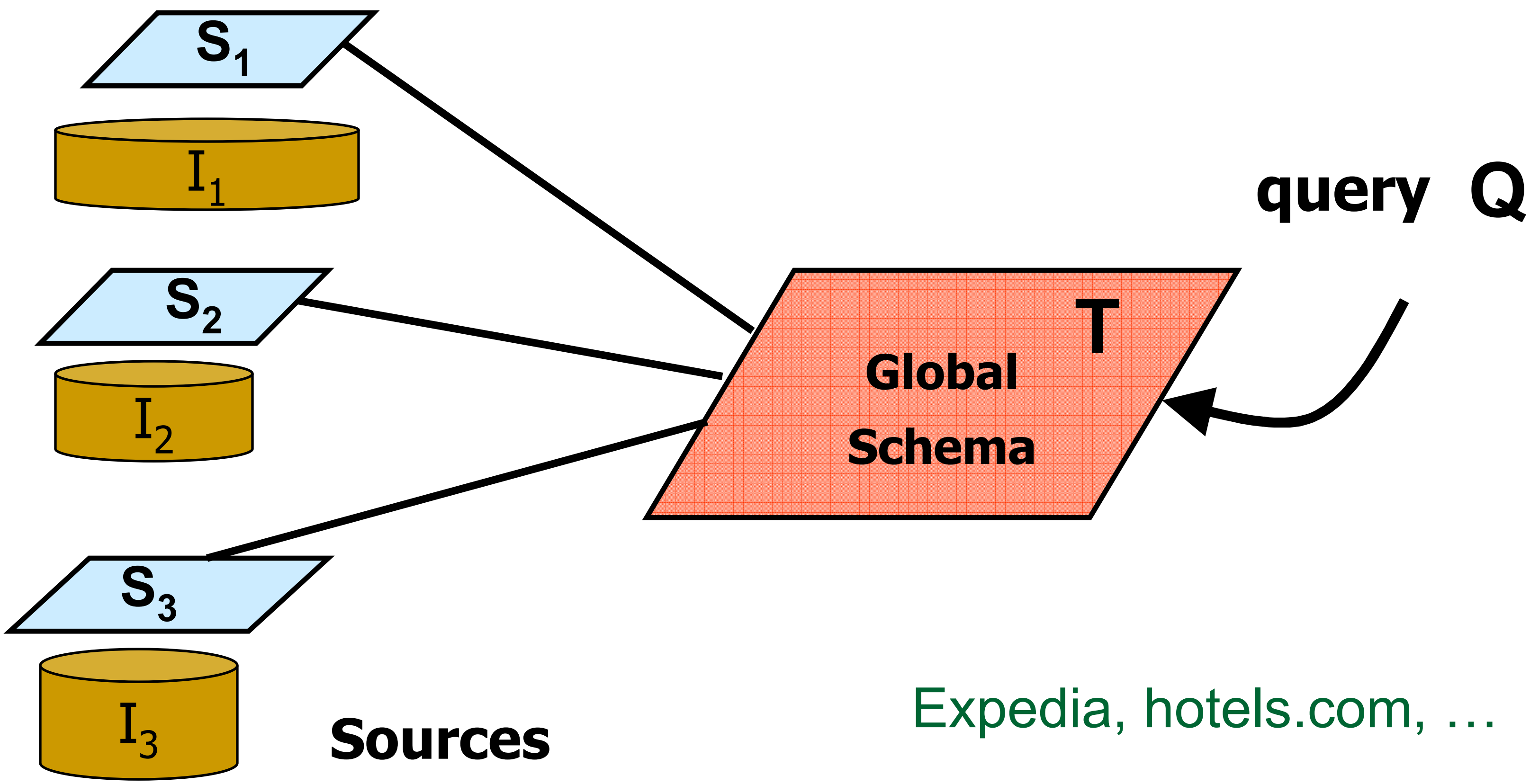
Two Facets of Information Integration

The research community has studied two different, but closely related, facets of information integration:

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

Data Integration

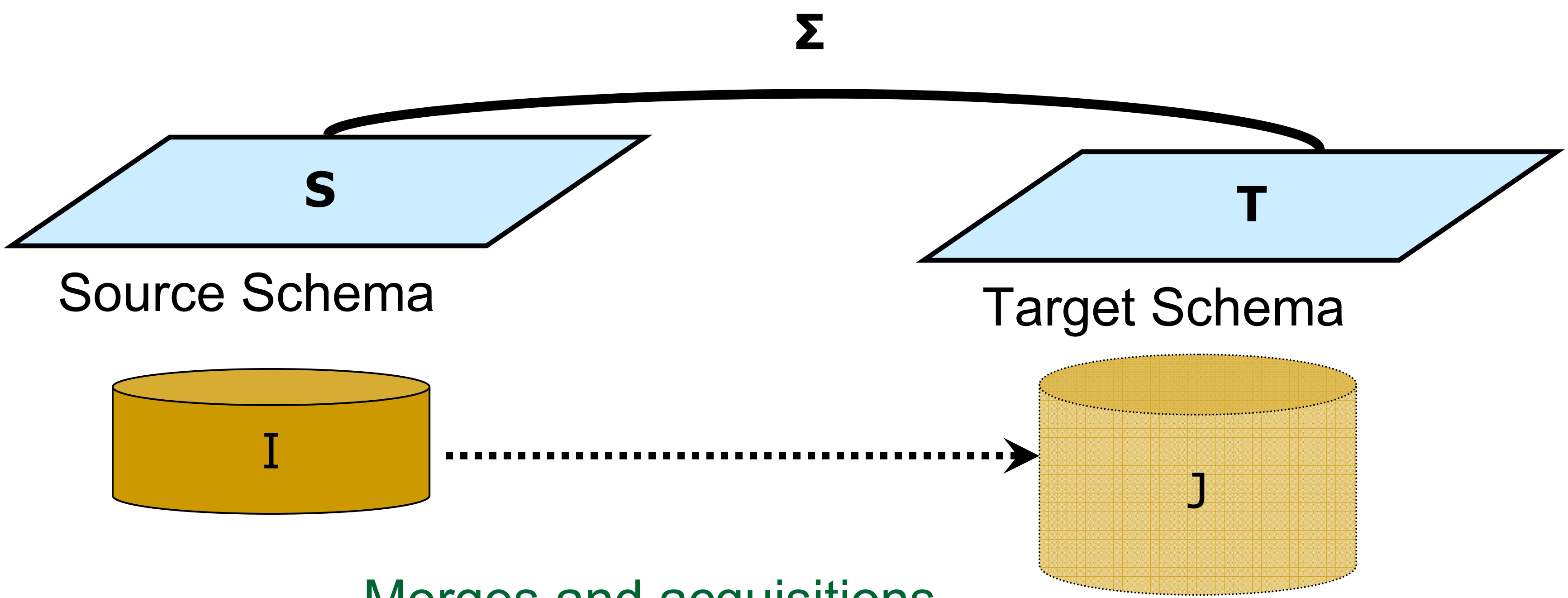
Query heterogeneous data in different sources via a virtual global schema



Expedia, hotels.com, …

Data Exchange

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



Merges and acquisitions, …

Schema Mappings

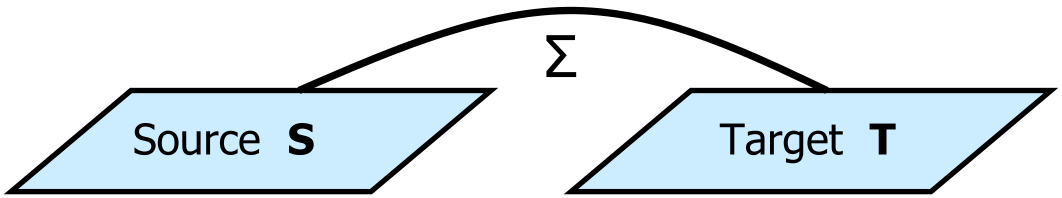
- Schema mappings constitute the essential building blocks in formalizing and studying data integration and data exchange.

- Schema mappings are:

  High-level, declarative assertions that specify the relationship between two database schemas.

- Schema mappings make it possible to separate the design of the relationship between schemas from its implementation.
  - Are easier to generate and manage (semi)-automatically;
  - 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**.

- Question: What is a "good" schema-mapping specification language?

Schema-Mapping Specification Languages

- **Obvious Idea**:
  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

Every schema-mapping specification language should support:

- Copy (Nicknaming):
  - Copy each source table to a target table and rename it.
- Projection (Column Deletion):
  - Form a target table by deleting one or more columns of a source table.
- Column Addition:
  - 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 addition+ …")

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

Schema-Mapping Specification Languages

- Question: What do all these tasks (copy, projection, column augmentation, decomposition, join) have in common?

- Answer:
  - They can be specified using tuple-generating dependencies (tgds).
  - In fact, they can be specified using a special class of tuple-generating dependencies known as source-to-target tuple generating dependencies (s-t tgds).

Database Integrity Constraints

- Dependency Theory: extensive study of integrity constraints in relational databases in the 1970s and 1980s
  (Codd, Fagin, Beeri, Vardi …)

- Tuple-generating dependencies (tgds) emerged as an important class of constraints with a balance between high expressive power and good algorithmic properties. Tgds are expressions of the form

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

  $\varphi(\mathbf{x})$, $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atomic formulas.

  Special Cases:
  - Inclusion Dependencies
  - Multivalued Dependencies

Tuple-Generating Dependencies

- "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!

Schema-Mapping Specification Language

The relationship between source and target is given by 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:

- $\forall s\ \forall c\ (\text{Student}\ (s) \wedge \text{Enrolls}(s,c) \rightarrow \exists g\ \text{Grade}(s,c,g))$
- (dropping the universal quantifiers in the front)

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

Schema-Mapping Specification Language

Fact: s-t tgds 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:

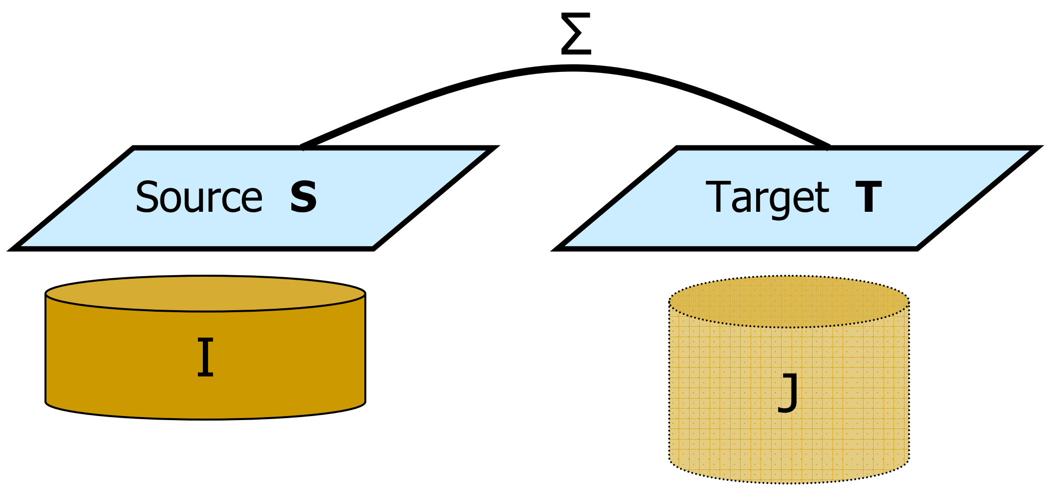
- Copy and projection
- Join: $\forall x \forall y \forall z\ (E(x,y) \wedge E(y,z) \rightarrow F(x,z))$

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

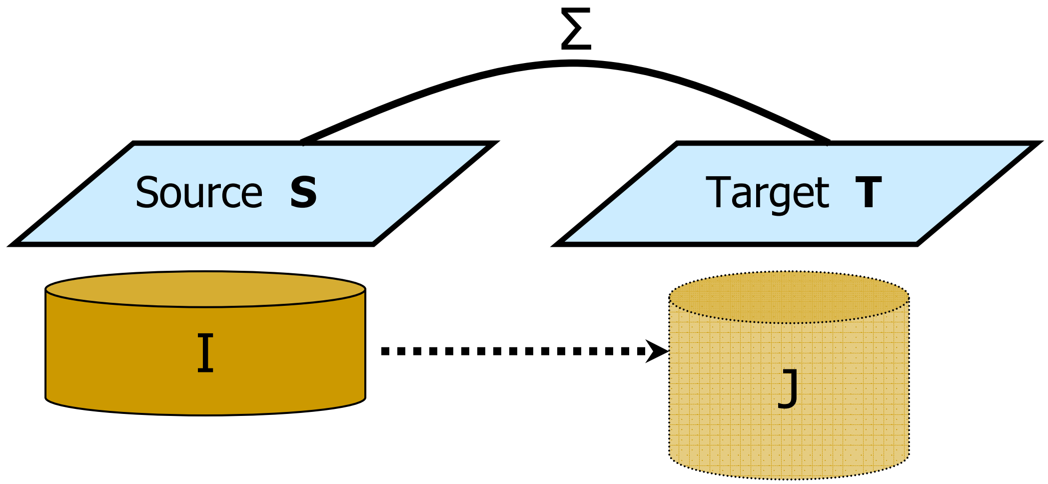
Semantics of Schema Mappings



- **M** = (**S**, **T**, Σ) schema mapping with Σ a set of s-t tgds
- From a semantic point of view, **M** can be identified with
  Inst(**M**) = { (I,J): I is a sourse instance,
  J is a target instance, and (I,J) $\models$ Σ }

  (this is open-world-assumption semantics)
- A **solution** for a source instance I is a target instance J such that (I,J) $\in$ Inst(**M**) (i.e., (I,J) $\models$ Σ).

Schema Mappings & Data Exchange



- **Data Exchange** via the schema mapping **M** = (**S**, **T**, Σ): Given a source instance I, construct a solution J for I.

- Difficulty:
  - Typically, there are multiple solutions
  - Which one is the "best" to materialize?

Over/Underspecification in Data Exchange

- **Fact:** A given source instance may have no solutions (overspecification)
- **Fact:** A given source instance may have multiple solutions (underspecification)

- **Example:**

  Source relation E(A,B), target relation H(A,B)

  Σ: $E(x,y) \rightarrow \exists z\, (H(x,z) \wedge H(z,y))$

  Source instance I = {E(a,b)}

  Solutions: Infinitely many solutions exist

  - J_1 = {H(a,b), H(b,b)}
  - J_2 = {H(a,a), H(a,b)}
  - J_3 = {H(a,X), H(X,b)}
  - J_4 = {H(a,X), H(X,b), H(a,Y), H(Y,b)}
  - J_5 = {H(a,X), H(X,b), H(Y,Y)}

constants:

a, b, …

variables (labelled nulls):

X, Y, …

Data Exchange & Universal solutions

Fagin, K …, Miller, Popa:

Identified and studied the concept of a **universal solution** in data exchange.

- A universal solution is a most general solution.
- A universal solution "represents" the entire space of solutions.
- A "**canonical**" universal solution can be generated efficiently using the **chase procedure**.

Universal Solutions in Data Exchange

Note: Two types of values in instances:

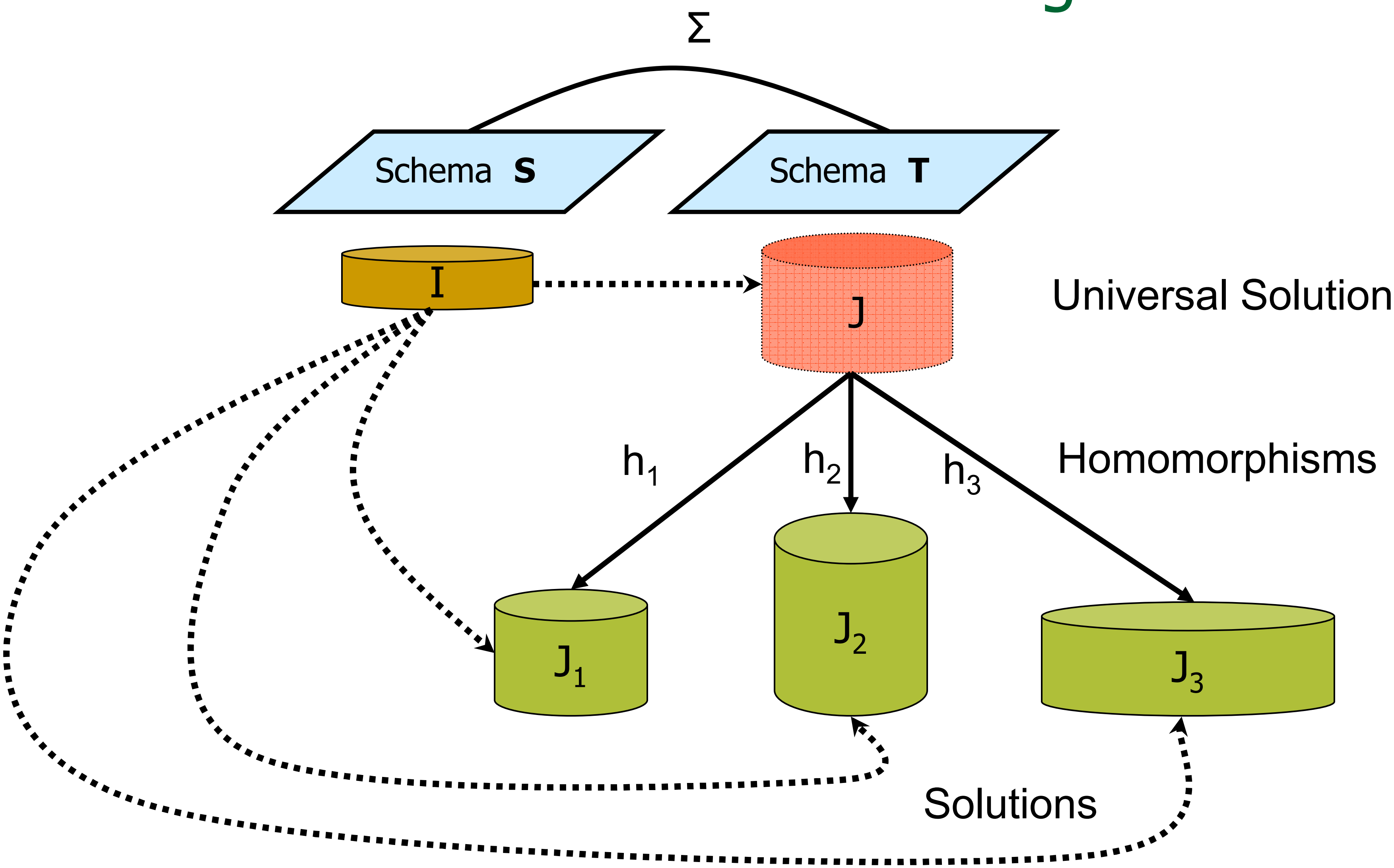
- **Constants**: they can only be mapped to themselves
- **Variables** (labeled nulls): they can be mapped to other values

Definition: Homomorphism $h: J \rightarrow K$ between instances:

- $h(c) = c$, for constant c
- If $P(a_1,\ldots,a_m)$ is in J, then $P(h(a_1),\ldots,h(a_m))$ is in K.

Definition (FKMP): A solution J for I is **universal** if it has homomorphisms to all other solutions for I.
(thus, a universal solution is a "most general" solution).

Universal Solutions in Data Exchange



Example

Source relation E(A,B), target relation F(A,B)

Σ : $E(x,y) \rightarrow \exists z\, (H(x,z) \wedge H(z,y))$

Source instance I = { E(1,2) }, where 1 and 2 are constants.

Solutions: Infinitely many solutions exist

- J_1 = { H(1,2), H(2,2) } is not universal
- J_2 = { H(1,1), H(1,2) } is not universal
- J_3 = { H(1,X), H(X,2) } is universal
- J_4 = { H(1,X), H(X,2), H(1,Y), H(Y,2) } is universal
- J_5 = { H(1,X), H(X,2), H(Y,Y) } is not universal

Structural Properties of Universal Solutions

- Universal solutions are akin to:
  - most general unifiers in logic programming;
  - initial models.

- Uniqueness up to homomorphic equivalence:

  If J and J' are universal for I, then they are homomorphically equivalent.

- Representation of the entire space of solutions:

  Assume that J is universal for I, and J' is universal for I'.

  Then the following are equivalent:

  1. I and I' have the same space of solutions.
  2. J and J' are homomorphically equivalent.

Exercise #1

- Let **M** = (**S**, **T**, Σ) be a GLAV schema mapping (i.e., Σ is a finite set of s-t tgds) and let I be a source instance. Assume that J is a universal solution for I, and J' is a universal solution for I'.

  Show that the following statements are equivalent:

  1. I and I' have the same space of solutions.
  2. J and J' are homomorphically equivalent.

- Does the above equivalence hold for schema mappings **M** = (**S**, **T**, Σ), where Σ is an arbitrary first-order sentence? Justify your answer as best as you can.

The Chase Procedure

Chase Procedure for **M** = (**S**, **T**, Σ) : given a source instance I, build a target instance $chase_M(I)$ 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 $chase_M(I)$ 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) LAV schema mapping with
Σ : $E(x,y) \rightarrow \exists z(F(x,z) \wedge F(z,y))$

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

- If I = { E(1,2) }, then $\text{chase}_{\mathbf{M}}(I)$ ={ E(1,X), E(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 : $E(x,z) \wedge E(z,y) \rightarrow F(x,y)$

- 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.

The Chase Procedure

Theorem (FKMP): Let **M**= (**S**, **T**, Σ) be a GLAV schema mapping (i.e., Σ is a set of s-t tgds). Then, for every source instance I,

- The chase procedure produces a universal solution $\text{chase}_{\mathbf{M}}(I)$.
- The running time of the chase procedure is bounded by a polynomial in the size of I (PTIME data complexity).

Note: The chase procedure can be used to produce universal solutions even in the presence of target constraints that obey certain mild structural conditions.

Target Dependencies

In addition to source-to-target dependencies, we also consider target dependencies:

- Target Tgds : $\varphi_T(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi_T(\mathbf{x}, \mathbf{y})$

  $Dpt\ (e,d) \rightarrow \exists p\ Proj(e,p)$

  (a target inclusion dependency constraint)

- Target Equality Generating Dependencies (egds):

  $\varphi_T(\mathbf{x}) \rightarrow (x_1 = x_2)$

  $Dpt\ (e, d_1) \wedge Dpt\ (e, d_2) \rightarrow (d_1 = d_2)$

  (a target key constraint)

Algorithmic Problems in Data Exchange

Question: Fix a schema mapping **M** = (**S**, **T**, Σ_{st},Σ_t) specified by s-t tgds and target tgds and egds. What can we say about the complexity of

- The existence-of-solutions problem **Sol(M)**
  (given source instance I, is there a solution for I w.r.t. **M**?

and

- The data exchange problem
  (given a source instance I, construct a universal solution for I w.r.t. **M**?)

Answer: Depending on the target constraints in Σ_t:

- **Sol(M)** is trivial (solutions always exist) /
  Universal solutions can be constructed in PTIME (in fact, in LOGSPACE).

  …
- **Sol(M)** can be in PTIME (in fact, it can be PTIME-complete) /
  Universal solutions can be constructed in PTIME (if solutions exist)

  …
- **Sol(M)** can be undecidable /
  Universal solutions may not exist (even if solutions exist)

Undecidability in Data Exchange

Theorem (K …, Panttaja, Tan):

There is a schema mapping **M**= (**S**, **T**, Σ^*_{st}, Σ^*_t) such that:

- Σ^*_{st} consists of a single s-t tgd;
- Σ^*_t consists of one target egd and two target tgds.
- The existence-of-solutions problem **Sol(M)** is undecidable.

Hint of Proof:

Reduction from the

Embedding Problem for Finite Semigroups

Given a finite partial semigroup, can it be embedded to a finite semigroup?

(**undecidability** implied by results of Evans and Gurevich).

The Embedding Problem & Data Exchange

Reducing the **Embedding Problem for Semigroups** to **Sol(M)**

- Σ_{st}: $R(x,y,z) \rightarrow R'(x,y,z)$
- Σ_t:
  - R′ is a partial function:
    $R'(x,y,z) \wedge R'(x,y,w) \rightarrow z = w$
  - R′ is associative
    $R'(x,y,u) \wedge R'(y,z,v) \wedge R'(u,z,w) \rightarrow R'(x,u,w)$
  - R′ is a total function
    $$R'(x,y,z) \wedge R'(x',y',z') \rightarrow \exists w_1 \ldots \exists w_9$$
    $$(R'(x,x',w_1) \wedge R'(x,y',w_2) \wedge R'(x,z',w_3)$$
    $$R'(y,x',w_4) \wedge R'(y,y',w_5) \wedge R'(x,z',w_6)$$
    $$R'(z,x',w_7) \wedge R'(z,y',w_8) \wedge R'(z,z',w_9))$$

Tractability in Data Exchange

Question: Are there broad structural conditions on the target constraints that guarantee tractability?
(that is,

- The existence of solutions problem is in PTIME

and

- A universal solution can be constructed in PTIME, if a solution exists.)

Algorithmic Properties of Universal Solutions

Theorem (FKMP): Schema mapping **M**= (**S**, **T**, Σ_{st}, Σ_t) such that:

- Σ_{st} is a set of source-to-target tgds;
- Σ_t is the union of a weakly acyclic set of target tgds with a set of target egds.

Then:

- Universal solutions exist if and only if solutions exist.
- **Sol(M)** is in PTIME.
- A *canonical* universal solution (if a solution exists) can be produced in PTIME using the chase procedure.

Chase Procedure for Tgds and Egds

Given a source instance I,

1. Use the naïve chase to chase I with Σ_{st} and obtain a target instance J*.

2. Chase J * with the target tgds and the target egds in Σ_t to obtain a target instance J as follows:

2.1. For target tgds introduce new facts in J as dictated by the RHS of the s-t tgd and introduce new values (variables) in J each time existential quantifiers need witnesses.

2.2. For target egds $\phi(x) \rightarrow x_1 = x_2$

2.2.1. If a variable is equated to a constant, replace the variable by that constant;

2.2.2. If one variable is equated to another variable, replace one variable by the other variable.

2.2.3 If one constant is equated to a different constant, stop and report "failure".

Weakly Acyclic Sets of Tgds: Definition

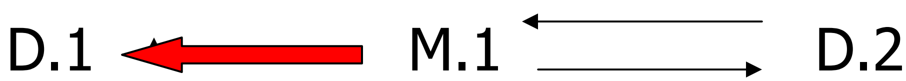
- **Position graph** of a set Σ of tgds:
  - **Nodes:** R.A, with R relation symbol, A attribute of R
  - **Edges:** for every $\varphi(\mathbf{x}) \to \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ in Σ, for every x in $\mathbf{x}$ occurring in ψ, for every occurrence of x in φ in R.A:
    - For every occurrence of x in ψ in S.B,
      add an edge R.A $\longrightarrow$ S.B
    - In addition, for every existentially quantified y that occurs in ψ in T.C, add a **special edge** R.A $\Longrightarrow$ T.C

- Σ is **weakly acyclic** if the position graph has **no** cycle containing a **special edge**.

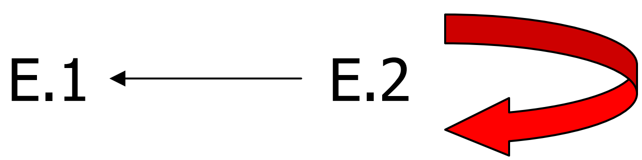
- A tgd θ is **weakly acyclic** if so is the singleton set $\{\theta\}$.

Weakly Acyclic Sets of Tgds: Examples

- **Example 1:** $\{\ D(e,m) \rightarrow M(m),\ \ M(m) \rightarrow \exists\, e\ D(e,m)\ \}$ is weakly acyclic, but cyclic.



- **Example 2:** $\{\ E(x,y) \rightarrow \exists\, z\ E(y,z)\ \}$ is not weakly acyclic.



Complexity of Data Exchange

$\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ Σ_{st} a set of s-t tgds	Existence-of-Solutions Problem	Existence-of-Universal Solutions Problem	Computing a Universal Solution
$\Sigma_t = \emptyset$; No target constraints	Trivial	Trivial	PTIME
Σ_t: Weakly acyclic set of target tgds + egds	PTIME It can be PTIME-complete	PTIME Univ. solutions exist if and only if solutions exist	PTIME
Σ_t: target tgds + egds	Undecidable, in general	Undecidable, in general	No algorithm exists, in general

Exercise #2

- Let **M**= (**S**, **T**, Σ) be a GLAV schema mapping (i.e., Σ is a set of s-t tgds). Show that the chase procedure for constructing universal solutions can be implemented in LOGSPACE.

- Let **M**= (**S**, **T**, Σ_{st}, Σ_t) be a schema mapping such that Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of target egds and a weakly acyclic set of target tgds.

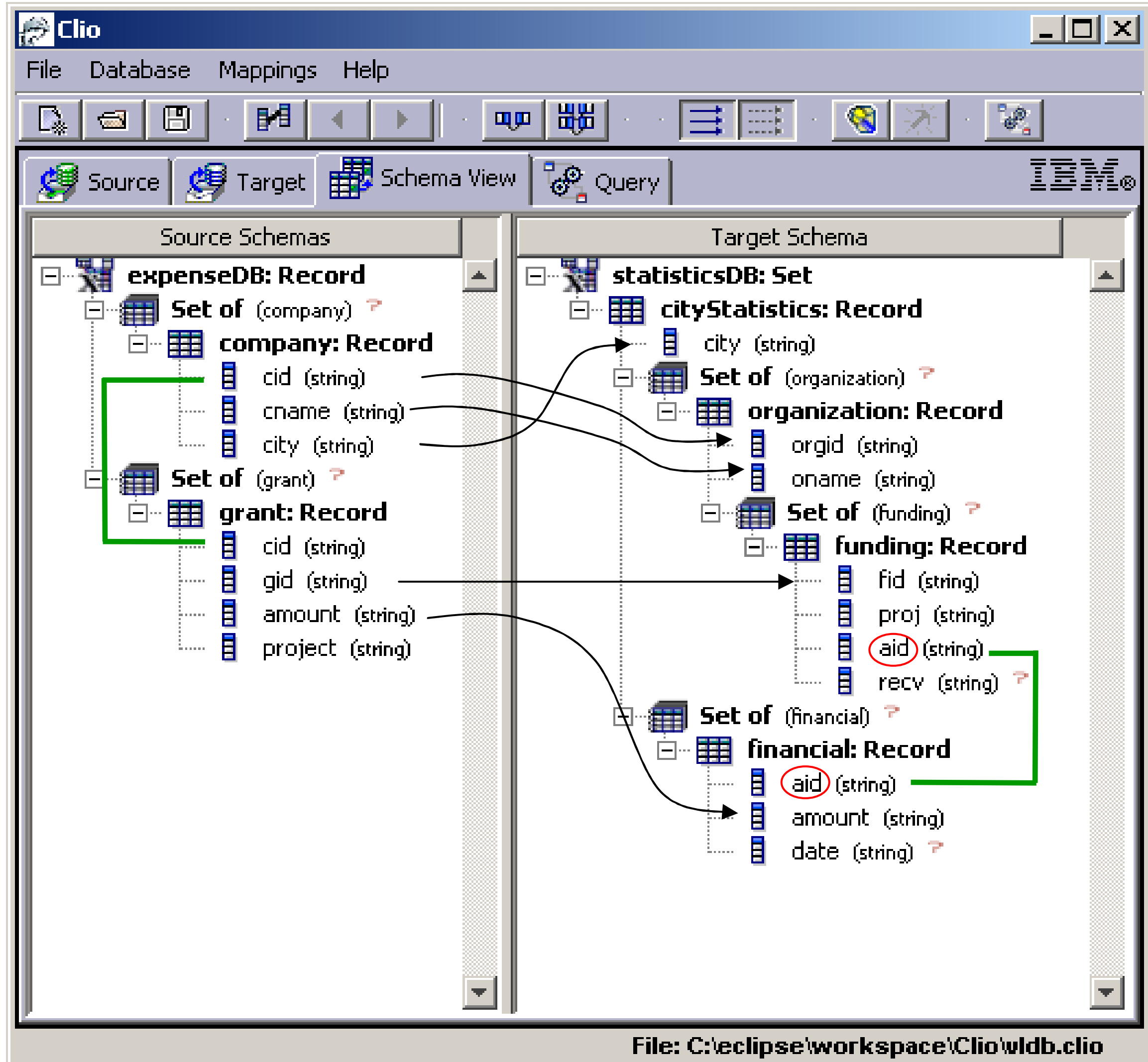
  Can the chase procedure be always implemented in LOGSPACE? Justify your answer as best as you can.

From Theory to Practice

- Clio Project at the IBM Almaden Research Center.
- Semi-automatic schema-mapping generation tool;
  - Data exchange system based on schema mappings.
- Universal solutions used as the semantics of data exchange.
- Universal solutions are generated via SQL queries extended with Skolem functions (implementation of chase procedure).
- Clio technology is now part of IBM Rational® Data Architect.

Some Features of Clio

- Supports nested structures
  - Nested Relational Model
  - Nested Constraints
- Automatic & semi-automatic discovery of attribute correspondence.
- Interactive derivation of schema mappings.
- Performs data exchange



The Smallest Universal Solution

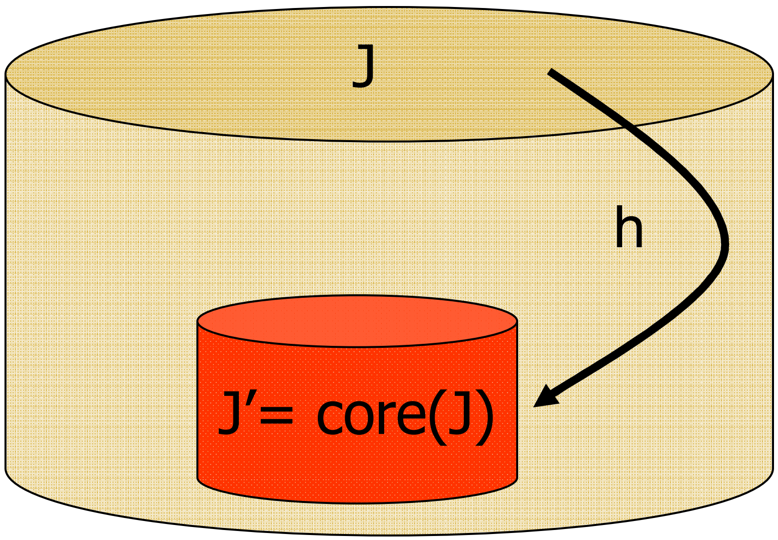
- **Fact:** Universal solutions need not be unique.
- **Question**: Is there a “best” universal solution?
- **Answer:** In joint work with R. Fagin and L. Popa, we took a “small is beautiful” approach:

  There is a smallest universal solution (if solutions exist); hence, the most compact one to materialize.

- **Definition:** The core of an instance J is the smallest subinstance J’ that is homomorphically equivalent to J.

- **Fact:**
  - Every finite relational structure has a core.
  - The core is unique up to isomorphism.

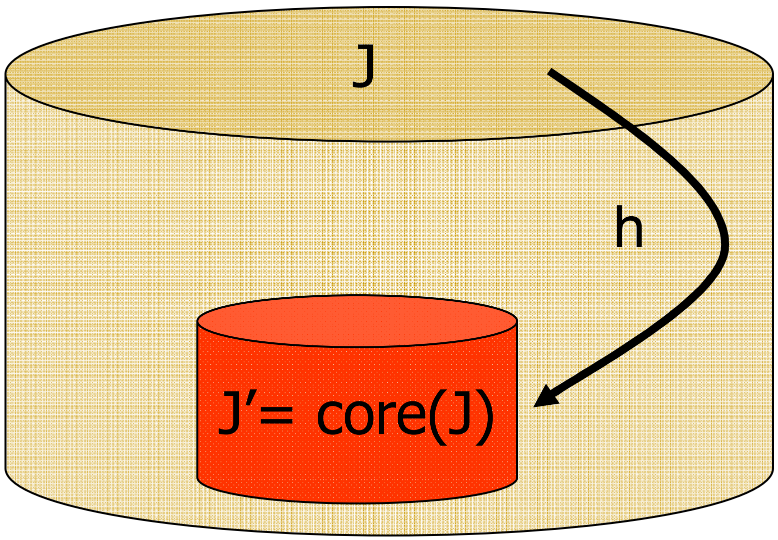
The Core of a Structure



Definition: J′ is the core of J if

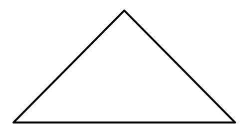
- $J' \subseteq J$
- there is a hom. $h: J \rightarrow J'$
- there is **no** hom. $g: J \rightarrow J''$, where $J'' \subset J'$.

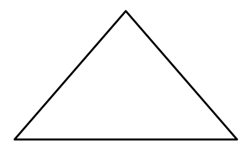
The Core of a Structure



Definition: J′ is the core of J if

- J′ $\subseteq$ J
- there is a hom. h: J $\rightarrow$ J′
- there is **no** hom. g: J $\rightarrow$ J″, where J″ $\subset$ J′.

Example: If a graph **G** contains a  , then

G is 3-colorable if and only if core(**G**) =  .

Fact: Computing cores of graphs is an NP-hard problem.

Example - continued

Source relation E(A,B), target relation H(A,B)

Σ : $(E(x,y) \rightarrow \exists z\, (H(x,z) \wedge H(z,y)))$

Source instance I = {E(a,b)}.

Solutions: Infinitely many universal solutions exist.

- J_3 = {H(a,X), H(X,b)} is the core.
- J_4 = {H(a,X), H(X,b), H(a,Y), H(Y,b)} is universal, but not the core.
- J_5 = {H(a,X), H(X,b), H(Y,Y)} is not universal.

Core: The smallest universal solution

Theorem (FKP 2003): **M** = (**S**, **T**, Σ_{st}, Σ_t) a schema mapping:

- All universal solutions have the same core.
- The core of the universal solutions is the smallest universal solution.
- If every target constraint is an egd, then the core is polynomial-time computable.

Theorem (Nash & Gottlob 2006): Let **M** = (**S**, **T**, Σ_{st}, Σ_t) be such that Σ_t is the union of a set of weakly acyclic target tgds with a set of target egds. Then the core is polynomial-time computable.

Exercise #3

- Prove that every finite graph **G** has a core.

- Prove that if both **H** and **H’** are cores of a finite graph G, then **H** and **H’** are isomorphic.

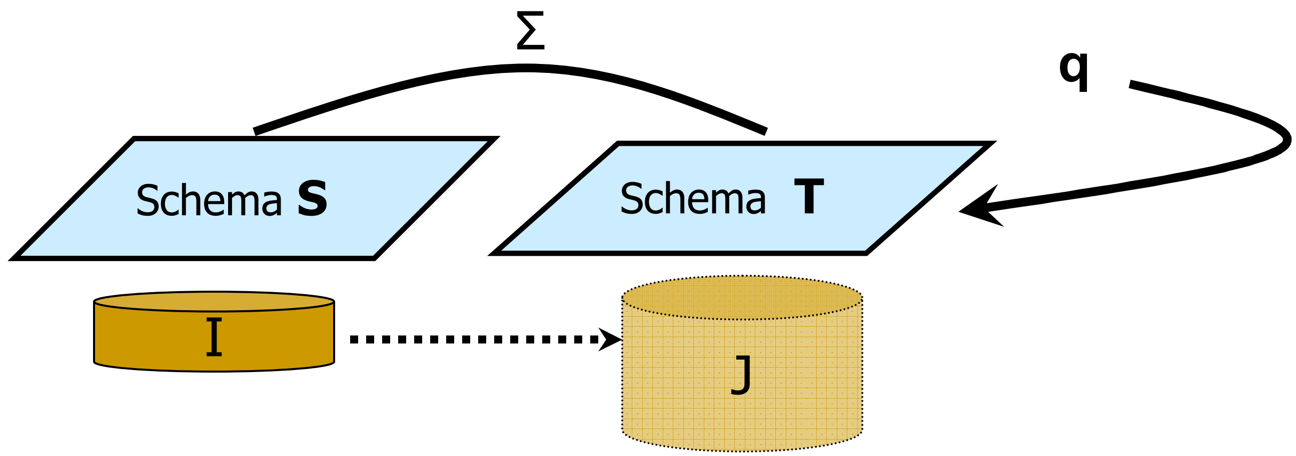
- Prove or disprove the following:
  Every infinite graph has a core.

- Identify the computational complexity of the
  CORE RECOGNITION PROBLEM:
  Given a finite graph **G**, is **G** its own core?

Outline of the Tutorial

- ✓ Schema Mappings as a framework for formalizing and studying data interoperability tasks.
- ✓ Data Exchange and Solutions in Data Exchange
  - Universal Solutions and the Core.
- Query Answering in Data Exchange.
- Managing schema mappings via operators:
  - The composition operator
  - The inverse operator and its variants

Query Answering in Data Exchange



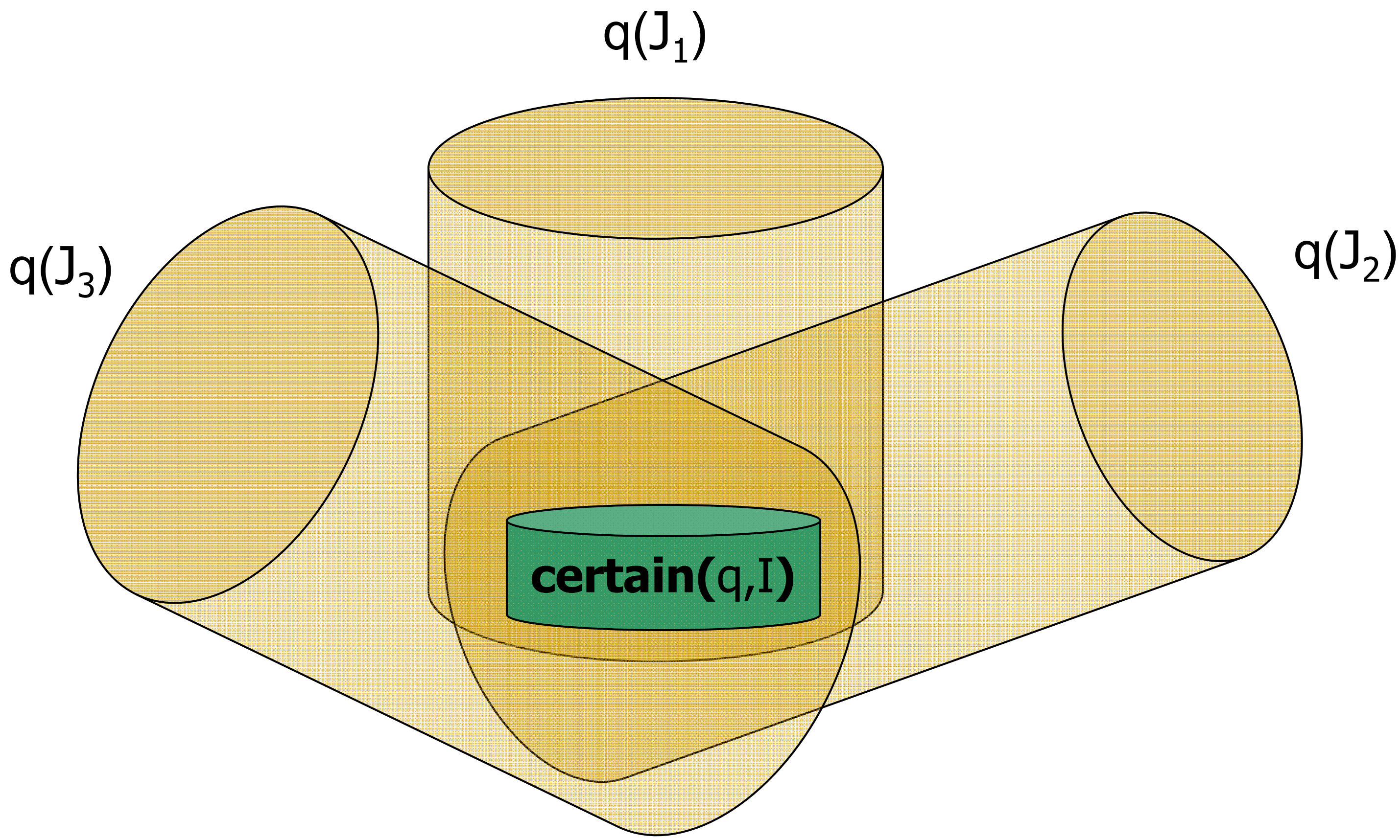
Question: What is the semantics of target query answering?

Definition: The certain answers of a query q over **T** on I

$$\textbf{certain}(q,I) = \bigcap \{\, q(J):\ J \text{ is a solution for } I \,\}.$$

Note: It is the standard open-world-assumption semantics in data integration.

Certain Answers Semantics



$$\textbf{certain}(q,I) = \bigcap \{\, q(J):\ J \text{ is a solution for } I \,\}.$$

Computing the Certain Answers

Theorem (FKMP): Schema mapping **M** = (**S**, **T**, Σ_{st}, Σ_t) such that:

- Σ_{st} is a set of source-to-target tgds, and
- Σ_t is the union of a weakly acyclic set of tgds with a set of egds.

Let q be a union of conjunctive queries over **T**.

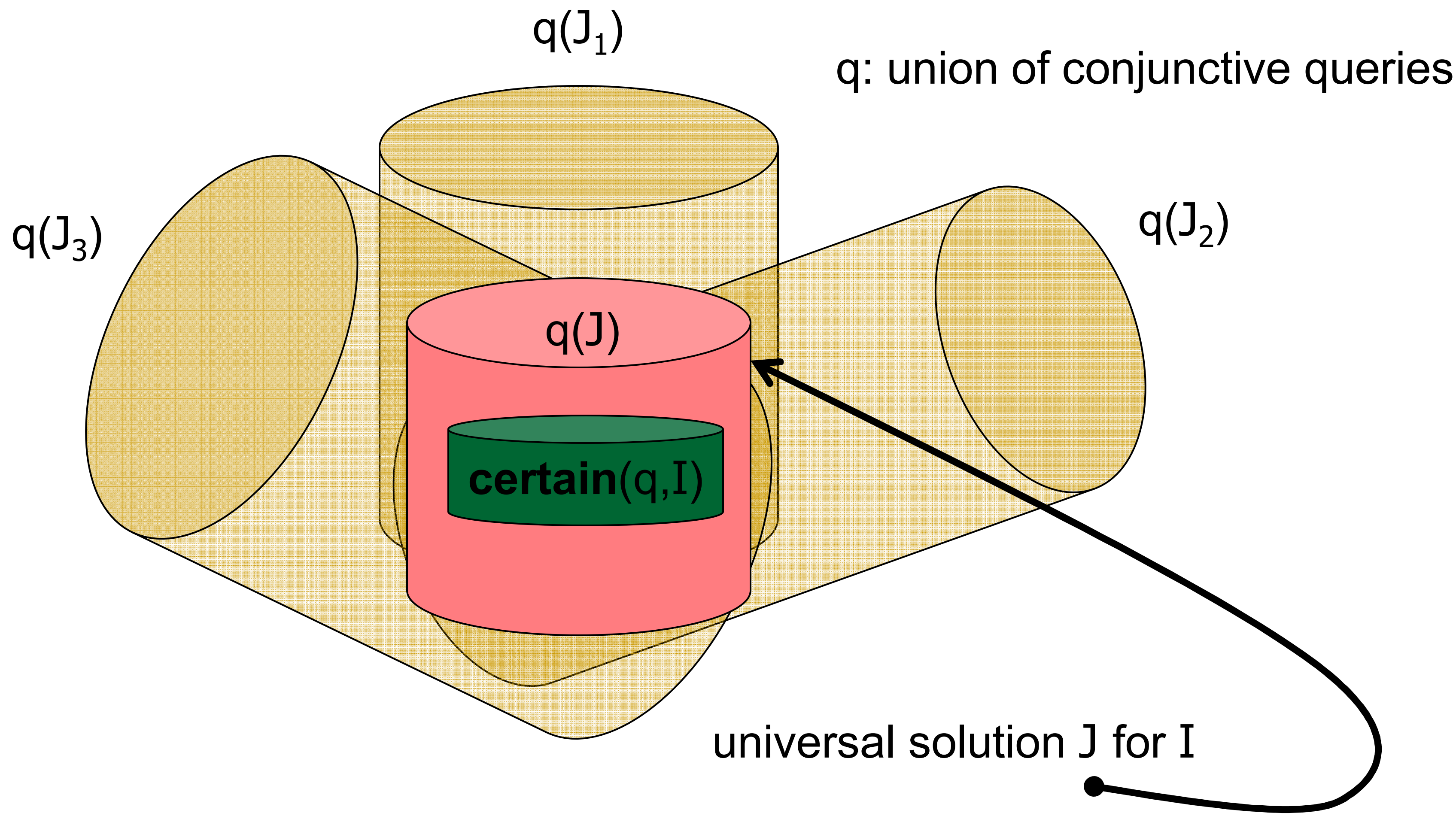
- If I is a source instance and J is a universal solution for I, then

  certain(q,I) = the set of all "null-free" tuples in q(J).

- Hence, **certain**(q,I) is computable in time polynomial in |I|:
  1. Compute a canonical universal J solution in polynomial time;
  2. Evaluate q(J) and remove tuples with nulls.

Note: This is a data complexity result (**M** and q are fixed).

Certain Answers via Universal Solutions



certain(q,I) = set of null-free tuples of q(J).

Computing the Certain Answers

Theorem (FKMP): Schema mapping **M** = (**S**, **T**, Σ_{st}, Σ_t) such that:

- Σ_{st} is a set of source-to-target tgds, and
- Σ_t is the union of a weakly acyclic set of tgds with a set of egds.

Let q be a union of conjunctive queries with inequalities ($\neq$).

- If q has at most one inequality per conjunct, then **certain**(q,I) is computable in time polynomial in |I| using a disjunctive chase.

- If q is has at most two inequalities per conjunct, then **certain**(q,I) can be coNP-complete, even if Σ_t is empty.

Outline of the Tutorial

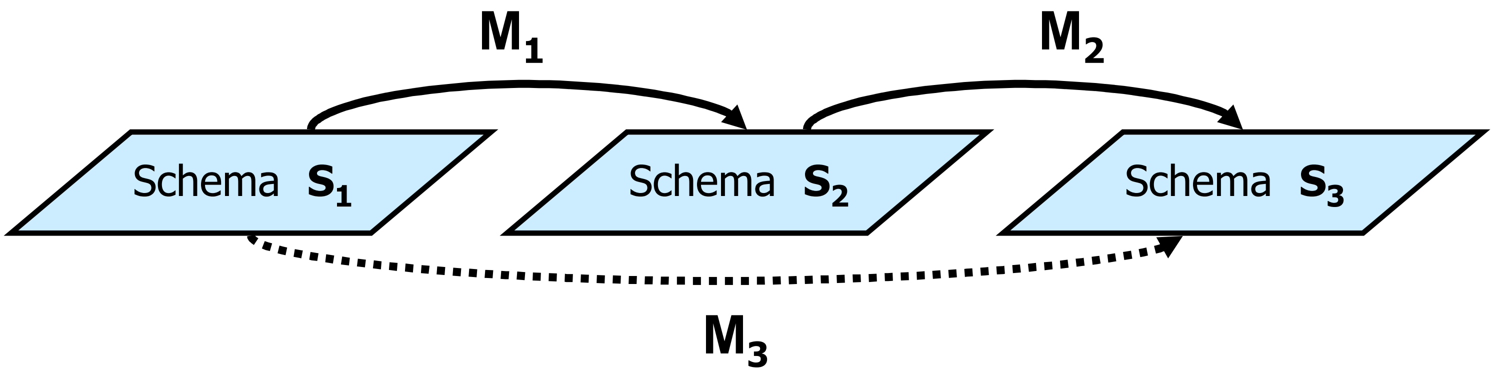
- Schema Mappings as a framework for formalizing and studying data interoperability tasks.
- Data Exchange and Solutions in Data Exchange
  - Universal Solutions and the Core.
- Query Answering in Data Exchange.
- Managing schema mappings via operators:
  - The composition operator
  - The inverse operator and its variants

Managing Schema Mappings

- Schema mappings can be quite complex.
- Methods and tools are needed to automate or semi-automate schema-mapping management.
- Metadata Management Framework – Bernstein 2003

  Based on schema-mapping operators, the most prominent of which are:
  - Composition operator
  - Inverse operator

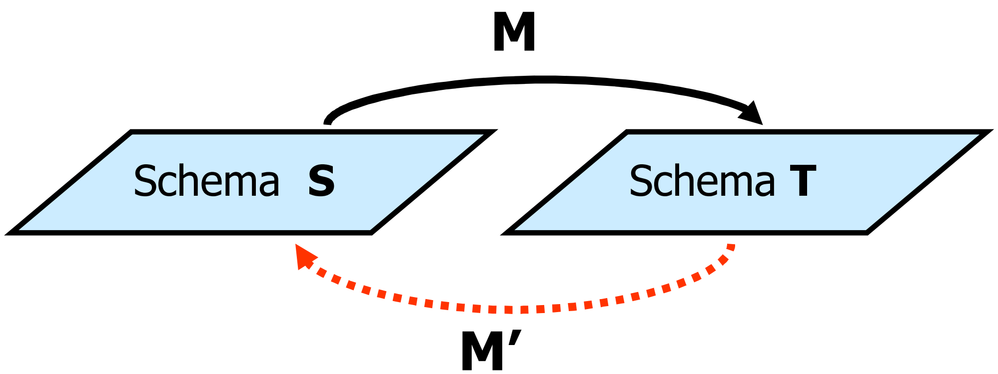
Composing Schema Mappings



- Given $\mathbf{M_1} = (\mathbf{S_1}, \mathbf{S_2}, \Sigma_1)$ and $\mathbf{M_2} = (\mathbf{S_2}, \mathbf{S_3}, \Sigma_2)$, derive a schema mapping $\mathbf{M_3} = (\mathbf{S_1}, \mathbf{S_3}, \Sigma_3)$ that is "equivalent" to the sequential application of $\mathbf{M_1}$ and $\mathbf{M_2}$.
- $\mathbf{M_3}$ is a **composition** of $\mathbf{M_1}$ and $\mathbf{M_2}$

$$\mathbf{M_3} = \mathbf{M_1} \circ \mathbf{M_2}$$

Inverting Schema Mapping

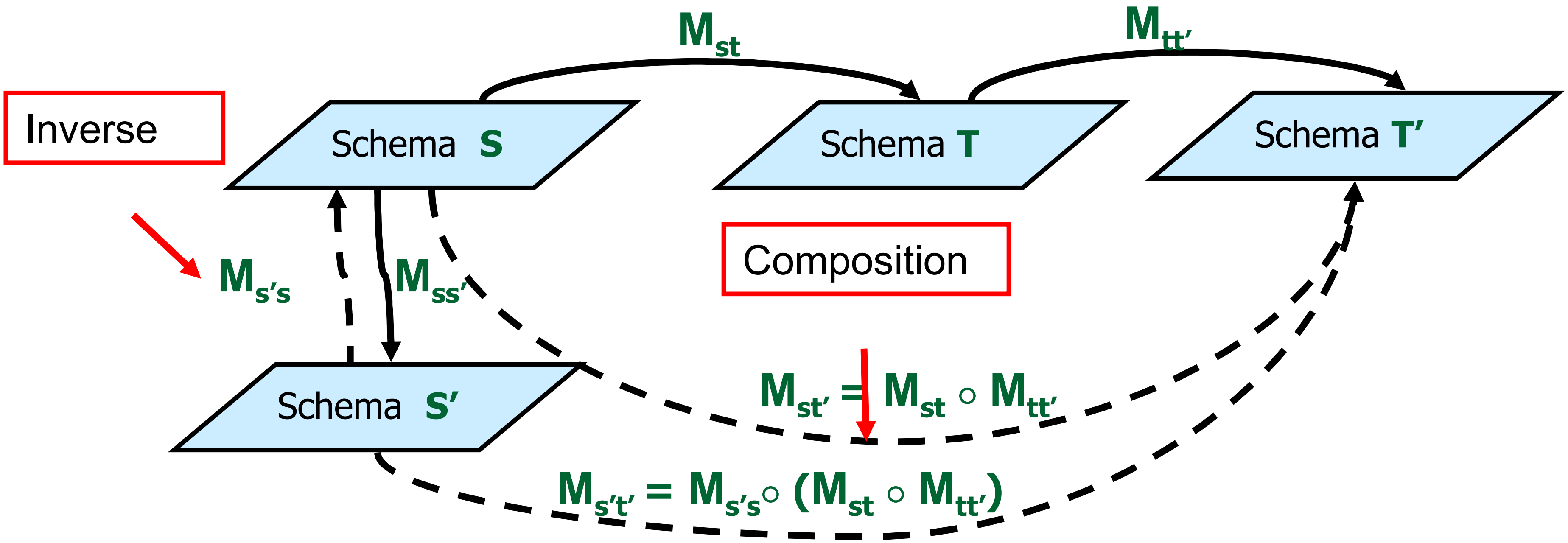


- Given **M**, derive **M'** that "undoes" **M**

  M' is an **inverse** of **M**

- Composition and inverse can be applied to schema evolution.

Applications to Schema Evolution



Fact:

Schema evolution can be analyzed using the composition operator and the inverse operator.

Composing Schema Mappings

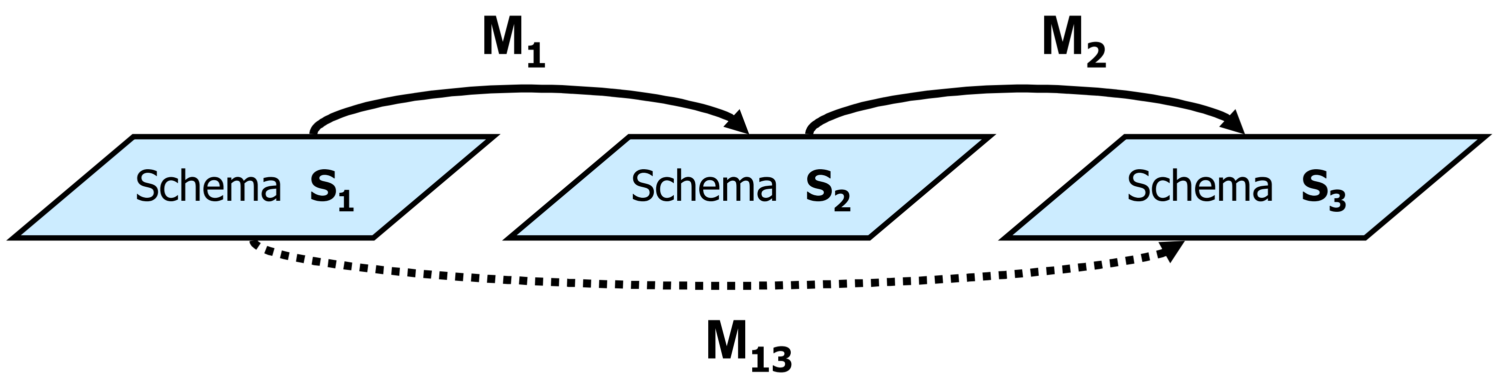
Main Issues:

- **Semantics**:
  What is the semantics of composition?

- **Language:**
  What is the language needed to express the composition of two schema mappings specified by s-t tgds?
  (GLAV schema mappings)

Note: Joint work with Fagin, Popa, and Tan

Composing Schema Mappings



- Given $\mathbf{M_1} = (\mathbf{S_1}, \mathbf{S_2}, \Sigma_1)$ and $\mathbf{M_2} = (\mathbf{S_2}, \mathbf{S_3}, \Sigma_2)$, derive a schema mapping $\mathbf{M_3} = (\mathbf{S_1}, \mathbf{S_3}, \Sigma_3)$ that is "equivalent" to the sequential application of $\mathbf{M_1}$ and $\mathbf{M_2}$.
- $\mathbf{M_3}$ is a **composition** of $\mathbf{M_1}$ and $\mathbf{M_2}$

$$\mathbf{M_3} = \mathbf{M_1} \circ \mathbf{M_2}$$

Semantics of Composition

- Recall that, from a semantic point of view, **M** can be identified with the binary relation

$$\mathrm{Inst}(\mathbf{M}) = \{\ (I,J):\ (I,J) \models \Sigma\ \}$$

- **Definition:**

A schema mapping $\mathbf{M_3}$ is a **composition** of $\mathbf{M_1}$ and $\mathbf{M_2}$ if

$$\mathrm{Inst}(\mathbf{M_3}) = \mathrm{Inst}(\mathbf{M_1}) \circ \mathrm{Inst}(\mathbf{M_2}), \text{ that is,}$$

$$(I_1,I_3) \models \Sigma_3$$

if and only if

there exists I_2 such that $(I_1,I_2) \models \Sigma_1$ and $(I_2,I_3) \models \Sigma_2$.

The Composition of Schema Mappings

Fact: If both $\mathbf{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma)$ and $\mathbf{M'} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma')$ are compositions of $\mathbf{M_1}$ and $\mathbf{M_2}$, then Σ are Σ' are logically equivalent. For this reason:

- We say that $\mathbf{M}$ (or $\mathbf{M'}$) is **the composition** of $\mathbf{M_1}$ and $\mathbf{M_2}$.
- We write $\mathbf{M_1} \circ \mathbf{M_2}$ to denote it

The Language of Composition: Good News

Theorem: Let $\mathbf{M_1}$ and $\mathbf{M_2}$ be consecutive schema mappings.

- If both $\mathbf{M_1}$ and $\mathbf{M_2}$ are GAV schema mappings, then their composition $\mathbf{M_1} \circ \mathbf{M_2}$ can be expressed as a GAV schema mapping.
- If $\mathbf{M_1}$ is a GAV schema mapping and $\mathbf{M_2}$ is a GLAV schema mappings, then their composition $\mathbf{M_1} \circ \mathbf{M_2}$ can be expressed as a GLAV schema mapping.

In symbols,

- GAV $\circ$ GAV = GAV
- GAV $\circ$ GLAV = GLAV

GAV $\circ$ GLAV = GLAV

Example:

- $\mathbf{M_1}$: GAV schema mapping

  $\text{Takes}(s,m,c) \rightarrow \text{Student}(s,m)$

  $\text{Takes}(s,m,c) \rightarrow \text{Enrolls}(s,c)$

- $\mathbf{M_2}$: GLAV schema mapping

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

- $\mathbf{M_1} \circ \mathbf{M_2}$: GLAV schema mapping

  $\text{Takes}(s,m,c) \wedge \text{Takes}(s,m',c') \rightarrow \exists g\ \text{Grade}(s,m,c',g)$

Exercise #4

- Show that
  GAV $\circ$ GAV = GAV

- Show that
  GAV $\circ$ GLAV = GLAV

- Give algorithms for
  - the composition GAV $\circ$ GAV

  and
  - the composition GAV $\circ$ GLAV.

- Analyze the running time of the algorithms you gave.

The Language of Composition: Bad News

Theorem:

- GLAV schema mappings are **not** closed under composition.

  In symbols, GLAV $\circ$ GLAV $\not\subseteq$ GLAV.

- In fact, there is a LAV schema mapping $\mathbf{M_1}$ and a GAV schema mapping $\mathbf{M_2}$ such that $\mathbf{M_1} \circ \mathbf{M_2}$ is **not** expressible in least fixed-point logic LFP (hence, not in FO or in Datalog).

  In symbols, LAV $\circ$ GAV $\not\subseteq$ LFP.

LAV $\circ$ GAV $\not\subset$ LFP

- **M$_1$** : LAV schema mapping
  $\forall x\ \forall y\ (E(x,y) \rightarrow \exists u \exists v\ (C(x,u) \wedge C(y,v)))$
  $\forall x\ \forall y\ (E(x,y) \rightarrow F(x,y))$
- **M$_2$** : GAV schema mapping
  $\forall x\ \forall y\ \forall u\ \forall v\ (C(x,u) \wedge C(y,v) \wedge F(x,y) \rightarrow D(u,v))$

- Given graph **G**=(V, E):
  - Let I_1 = E
  - Let I_3 = { D(r,g), D(g,r), D(b,r), D(r,b), D(g,b), D(b,g) }

  Fact:
  G is 3-colorable if and only if $(I_1, I_3) \in$ Inst(**M$_1$**) $\circ$ Inst(**M$_2$**)

- **Theorem (Dawar – 1998):**
  3-Colorability is **not** expressible in LFP.

The Language of Composition

Question:

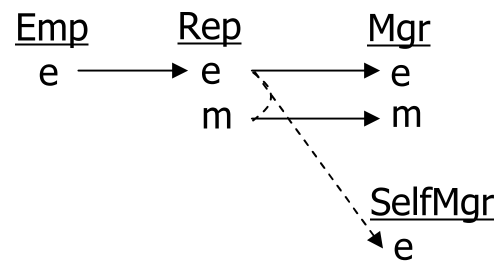
What is the “right” language for expressing the composition of two GLAV schema mappings?

Answer:

A fragment of **existential second-order logic** turns out to be the “right” language for this task.

Second-Order Logic to the Rescue

- $\mathbf{M_1}$: LAV schema mapping
  - $\forall e\ (Emp(e) \rightarrow \exists m\ Rep(e,m))$
- $\mathbf{M_2}$: GAV schema mapping
  - $\forall e\ \forall m\ (Rep(e,m) \rightarrow Mgr(e,m))$
  - $\forall e\ (Rep(e,e) \rightarrow SelfMgr(e))$



- **Theorem:** $\mathbf{M_1} \circ \mathbf{M_2}$ is **not** definable by **any** set (finite or infinite) of s-t tgds.

- **Fact**: This composition is definable in a well-behaved fragment of existential second-order logic, called SO tgds, that extends s-t tgds with Skolem functions.

Second-Order Logic to the Rescue

- $\mathbf{M_1}$: LAV schema mapping
  - $\forall e\ (Emp(e) \rightarrow \exists m\ Rep(e,m))$
- $\mathbf{M_2}$: GAV schema mapping
  - $\forall e\ \forall m\ (Rep(e,m) \rightarrow Mgr(e,m))$
  - $\forall e\ (Rep(e,e) \rightarrow SelfMgr(e))$

- **Fact:** $\mathbf{M_1} \circ \mathbf{M_2}$ is expressible by the SO-tgd
  - $\exists \mathbf{f}\ (\forall e\ (Emp(e) \rightarrow Mgr(e,\mathbf{f(e)})) \wedge$
    $\forall e\ (Emp(e) \wedge (\mathbf{e=f(e)}) \rightarrow SelfMgr(e)))$.

Second-Order Tgds

Definition: Let **S** be a source schema and **T** a target schema.

A second-order tuple-generating dependency (SO tgd) is a formula of the form:

$\exists f_1 \ldots \exists f_m((\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)) \wedge \ldots \wedge (\forall \mathbf{x}_n(\phi_n \rightarrow \psi_n))),$ where

- Each f_i is a function symbol.
- Each ϕ_i is a conjunction of atoms from **S** and equalities of terms.
- Each ψ_i is a conjunction of atoms from **T.**

Example: $\exists \mathbf{f}\ (\forall e(\ Emp(e) \rightarrow Mgr(e,\mathbf{f(e)})\ ) \wedge$
$\forall e(\ Emp(e) \wedge (\mathbf{e=f(e)}) \rightarrow SelfMgr(e)\ )\ )$

Composing SO-Tgds and Data Exchange

Theorem (FKPT):

- The composition of two SO-tgds is definable by a SO-tgd.
- There is an algorithm for composing SO-tgds.
- The chase procedure can be extended to SO-tgds; it produces universal solutions in polynomial time.
- Every SO tgd is the composition of finitely many GLAV schema mappings. Hence, SO tgds are the "right" language for the composition of GLAV schema mappings.

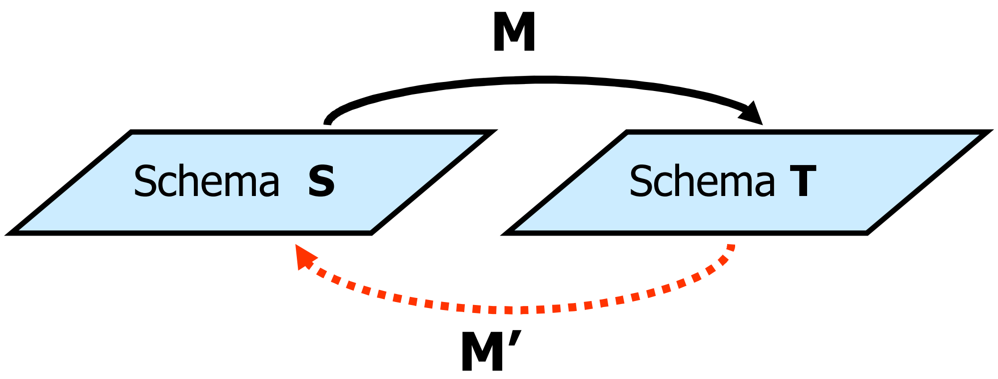
Synopsis of Schema Mapping Composition

- GAV $\circ$ GAV = GAV
- GAV $\circ$ GLAV = GLAV.
- GLAV $\circ$ GLAV $\not\subseteq$ GLAV. In fact, LAV $\circ$ GAV $\not\subseteq$ LFP.
- GLAV $\circ$ GLAV = SO-tgds = SO-tgds $\circ$ SO-tgds
  - SO-tgds are the "right" language for composing GLAV schema mappings.
  - SO-tgds are "chasable": Universal solutions in PTIME.
  - SO-tgds and the composition algorithm are supported in Clio.

Related Work (partial list)

- Earlier work on composition
  Madhavan and Halevy - 2003
- Composing richer schema mappings
  Nash, Bernstein, Melnik – 2007
- Composing schema mappings in open & closed worlds
  Libkin and Sirangelo – 2008
- XML Schema Mappings
  Amano, Libkin, Murlak – 2009
- Composing schema mappings with target constraints
  Arenas, Fagin, Nash – 2010
- Composing LAV schema mappings with distinct variables
  Arocena, Fuxman, Miller - 2010

Inverting Schema Mapping



- Given **M**, derive **M’** that “undoes” **M**.
- **Question:**
  What is the “right” semantics of the inverse operator?
- **Note:**
  In general, **M** may have no “good” inverse, because **M** may have **information loss** (e.g., projection schema mapping).

The Semantics of the Inverse Operator

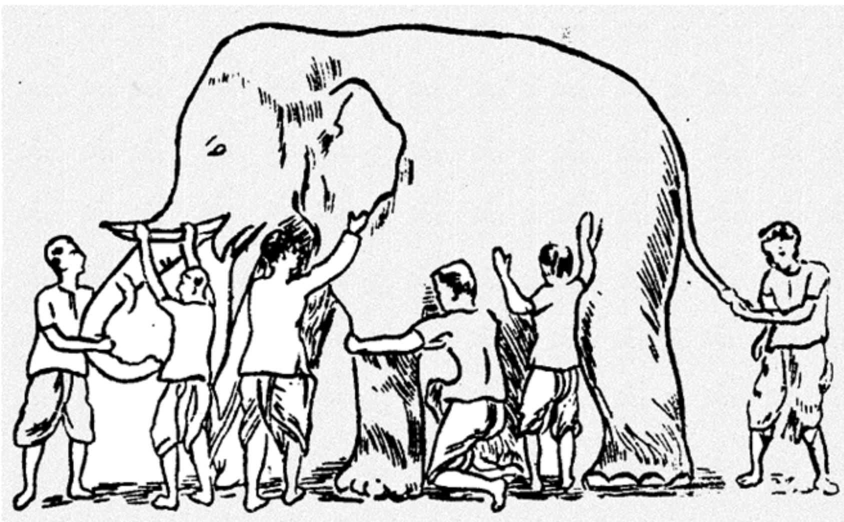
- Several different approaches:
  - (Exact) Inverses of schema mappings
    Fagin - 2006
  - Quasi-inverses of schema mappings
    Fagin, K …, Popa, Tan - 2007
  - Maximum recoveries of schema mappings
    Arenas, Pérez, Riveros - 2008
  - Extended maximum recoveries of schema mappings
    Fagin, K …, Popa, Tan – 2009

- **No** definitive semantics of the inverse operator has emerged.

Related Presentations at DEIS '10

- Adrian Onet: Chase and its Applications to Data Exchange
- Vadim Savenkov: Core Computation for Data Exchange
- Jorge Pérez: Inverting Schema Mappings
- André Hernich: Closed World Reasoning in Data Exchange
- Amelie Greebranddt: XML Data Exchange
- Víctor Gutiérrez-Basulto: Integrity Constraints in Data Exchange
- Emanuel Salinger: Analyzing, Comparing, and Debugging Schema Mappings

Data Interoperability: The Elephant and the Six Blind Men



- Data interoperability remains a major challenge:

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

- Schema mappings specified by tgds offer a formalism that covers only some aspects of data interoperability.

- However, theory and practice can inform each other.