Foundations and Applications
of
Schema Mappings

Phokion G. Kolaitis

University of California Santa Cruz
&
IBM Almaden Research Center

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:
  - $1.44B in 2007; 17% annual rate of growth
  - 15 major vendors in Gartner’s Magic Quadrant Report (source: Gartner, Inc., September 2008)

Gartner’s Magic Quadrant Report on Data Interoperability Products

Ability to execute

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

Completeness of vision

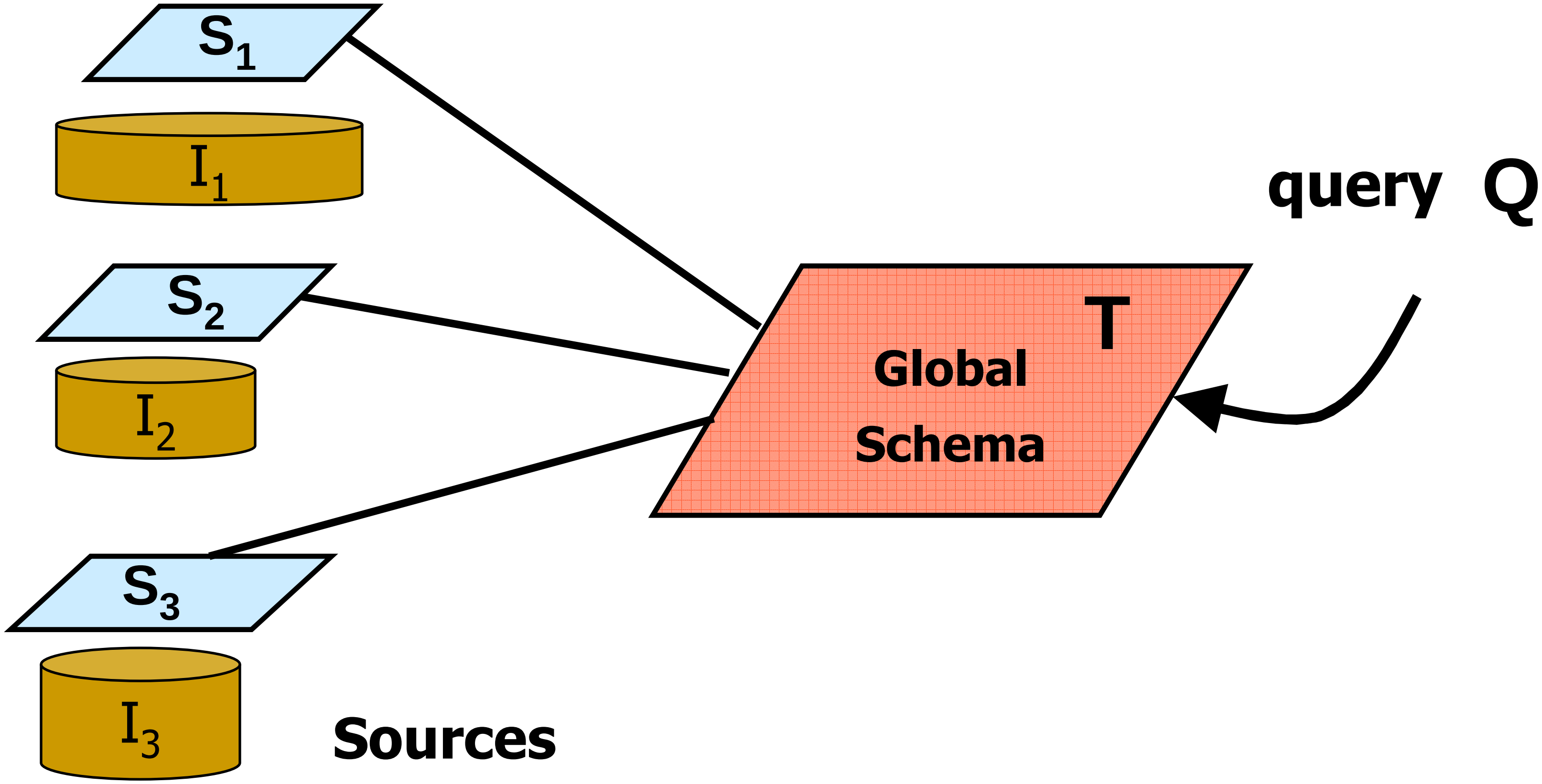
Theoretical Aspects of Data Interoperability

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

- **Data Integration** (aka **Data Federation**)
  - Formalized and studied for the past 10-15 years
- **Data Exchange** (aka **Data Translation**)
  - Formalized and studied for the past 5 years

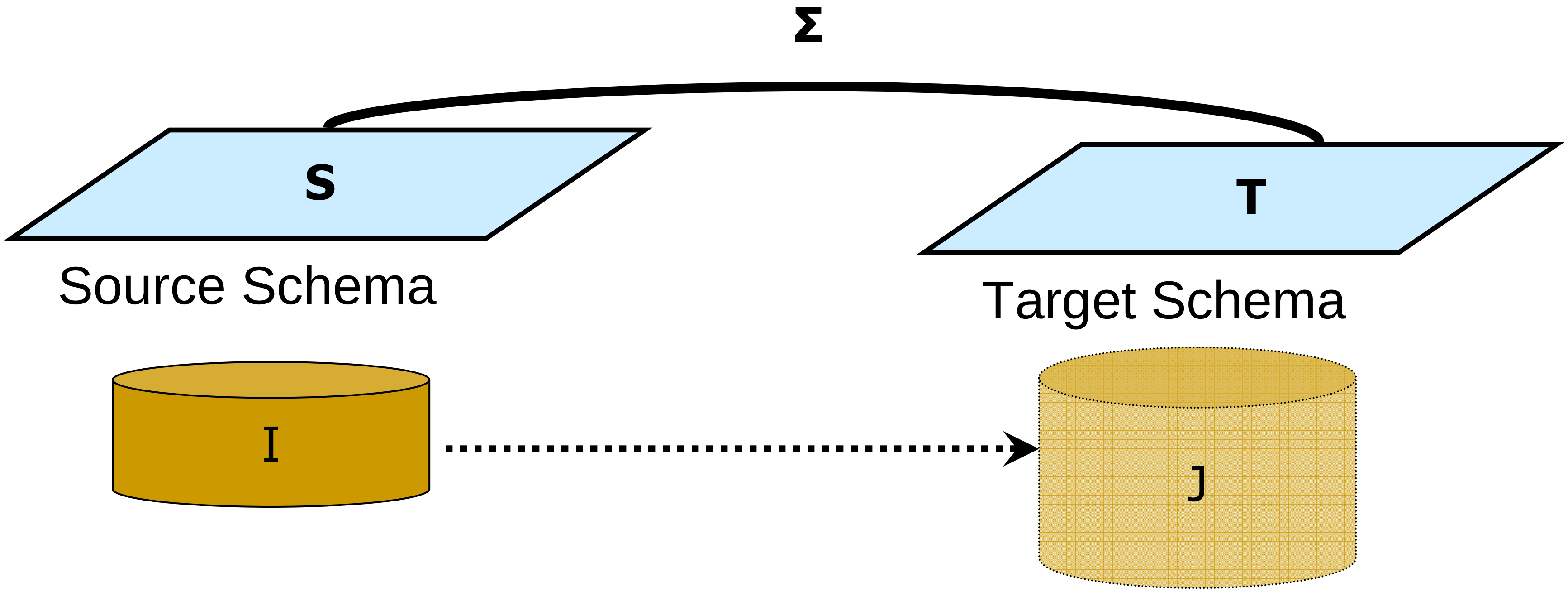
Data Integration

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.



Data Exchange

Data Exchange is an old, but recurrent, database problem

- Phil Bernstein – 2003
  "*Data exchange is the oldest database problem*"

- **EXPRESS**: IBM San Jose Research Lab – 1977
  EXtraction, **P**rocessing, and **RES**tructuring **S**ystem
  for transforming data between hierarchical databases.

- Data Exchange underlies several data interoperability tasks:
  - XML Publishing, XML Storage, …
  - Data Warehousing, ETL (Extract-Transform-Load).

The Data Interoperability Challenge

Fact:

- Data interoperability tasks require expertise, effort, and time.
- In particular, human experts have to generate complex transformations that specify the relationship between schemas written as programs (e.g., in Java) or as SQL/XSLT scripts.
- At present, there is relatively little automation in this area.

Question: How can we do better than this?

Answer: Introduce a higher level of abstraction that makes it possible to separate the design of the relationship between schemas from its implementation.

Schema Mappings

- Schema mappings:

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

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

- Schema mappings help with the development of tools:
  - Are easier to generate and manage (semi)-automatically;
  - Can be compiled into SQL/XSLT scripts automatically.

Outline

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

- Schema Mappings and Data Exchange
  - Algorithmic problems in data exchange.
  - Solutions, universal solutions, and the core.

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

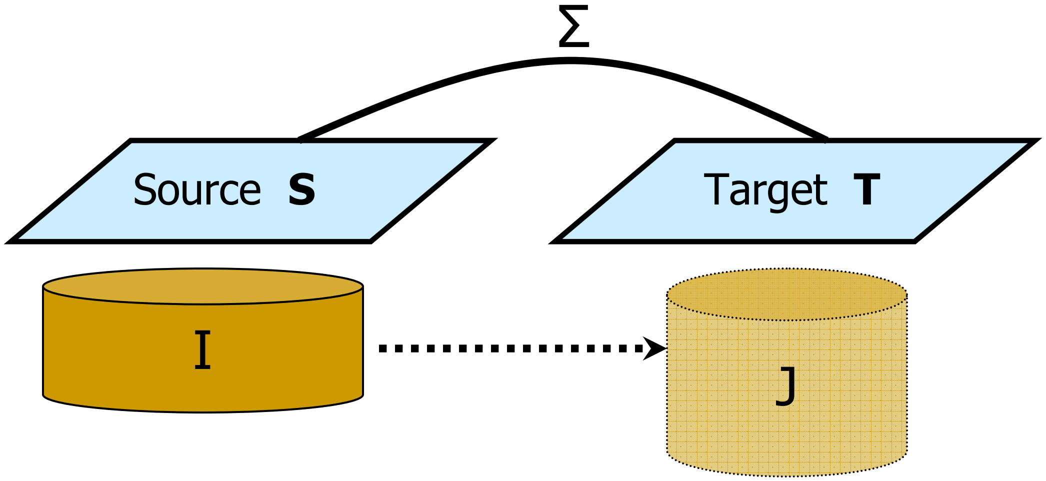
Acknowledgments

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

  Papers in ICDT 2003, PODS 2003-2008, 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.

Schema Mappings & Data Exchange



- 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**.
- Data Exchange via the schema mapping **M** = (**S**, **T**, Σ)

  Transform a given source instance I to a target instance J, so that (I, J) satisfy the specifications Σ of **M**.

Schema Mapping Specification Languages

- Ideally, schema mappings should be
  - expressive enough to specify data interoperability tasks;
  - simple enough to be efficiently manipulated by tools.
- **Question**: How are schema mappings specified?
- **Answer:** Use a suitable logical formalism.
- **Warning:** Unrestricted use of first-order logic as a schema mapping specification language gives rise to **undecidability** phenomena.

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

- 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 …)

- Two main classes of constraints with a balance between high expressive power and good algorithmic properties:
  - Tuple-generating dependencies (tgds)

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

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

  - Equality-generating dependencies (egds)

    $\forall \mathbf{x}\, (\varphi(\mathbf{x}) \rightarrow (x_i = x_j))$

    Special Case: Functional dependencies (in particular, keys)

    $\forall x, y, z\, (\mathrm{Manages}(x,z) \wedge \mathrm{Manages}(y,z) \rightarrow (x = y))$

Schema Mapping Specification Language

The relationship between source and target is given by source-to-target tuple generating dependencies (s-t tgds)

$$\forall \mathbf{x}\,(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\,\psi(\mathbf{x}, \mathbf{y})),$$ 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: (dropping the universal quantifiers in the front)

- $(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow \exists t\, \exists g\, (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$
- $E(x,y) \wedge E(y,z) \rightarrow F(x,z)$ (GAV (full) constraint)
- $E(x,y) \rightarrow \exists\, z\, (H(x,z) \wedge H(z,y))$ (LAV constraint)

Target Dependencies

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

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

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

  (a target inclusion dependency constraint)

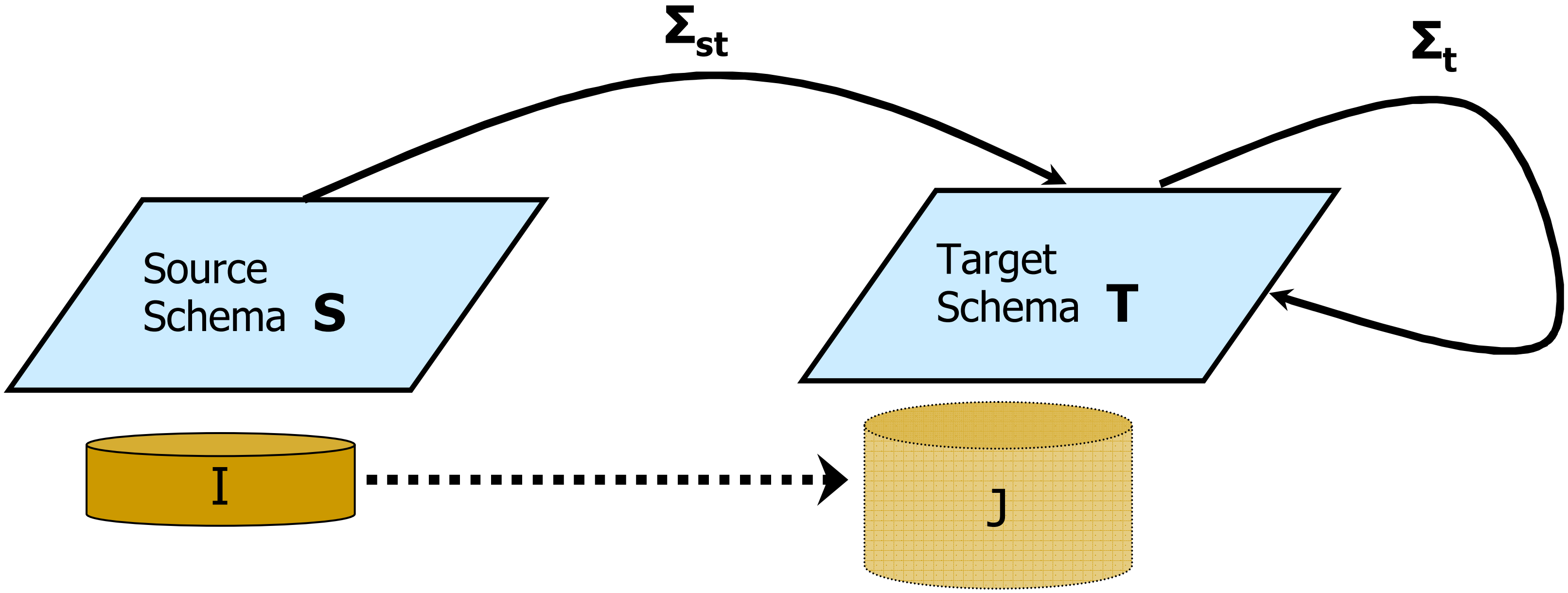
- Target Equality Generating Dependencies (egds):

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

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

  (a target key constraint)

Data Exchange Framework



Schema Mapping **M** = (**S**, **T**, Σ_{st} , Σ_t), where

- Σ_{st} is a set of source-to-target tgds
- Σ_t is a set of target tgds and target egds

Algorithmic Problems in Data Exchange

Definition: Schema Mapping **M** = (**S**, **T**, Σ_{st} , Σ_t)

- A target instance J is a solution for a source instance I if

$$(I,J) \models \Sigma_{st} \cup \Sigma_t .$$

- The existence-of-solutions problem **Sol(M)**: (decision problem)
  Given a source instance I, is there a solution J for I?

- The data exchange problem associated with **M**: (function problem)
  Given a source instance I, construct a solution J for I, provided a solution exists.

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, …

Main issues in data exchange

For a given source instance, there may be multiple target instances satisfying the specifications of the schema mapping. Thus,

- When more than one solution exist, which solutions are "better" than others?
- How do we compute a "best" solution?
- In other words, what is the "right" semantics of data exchange?

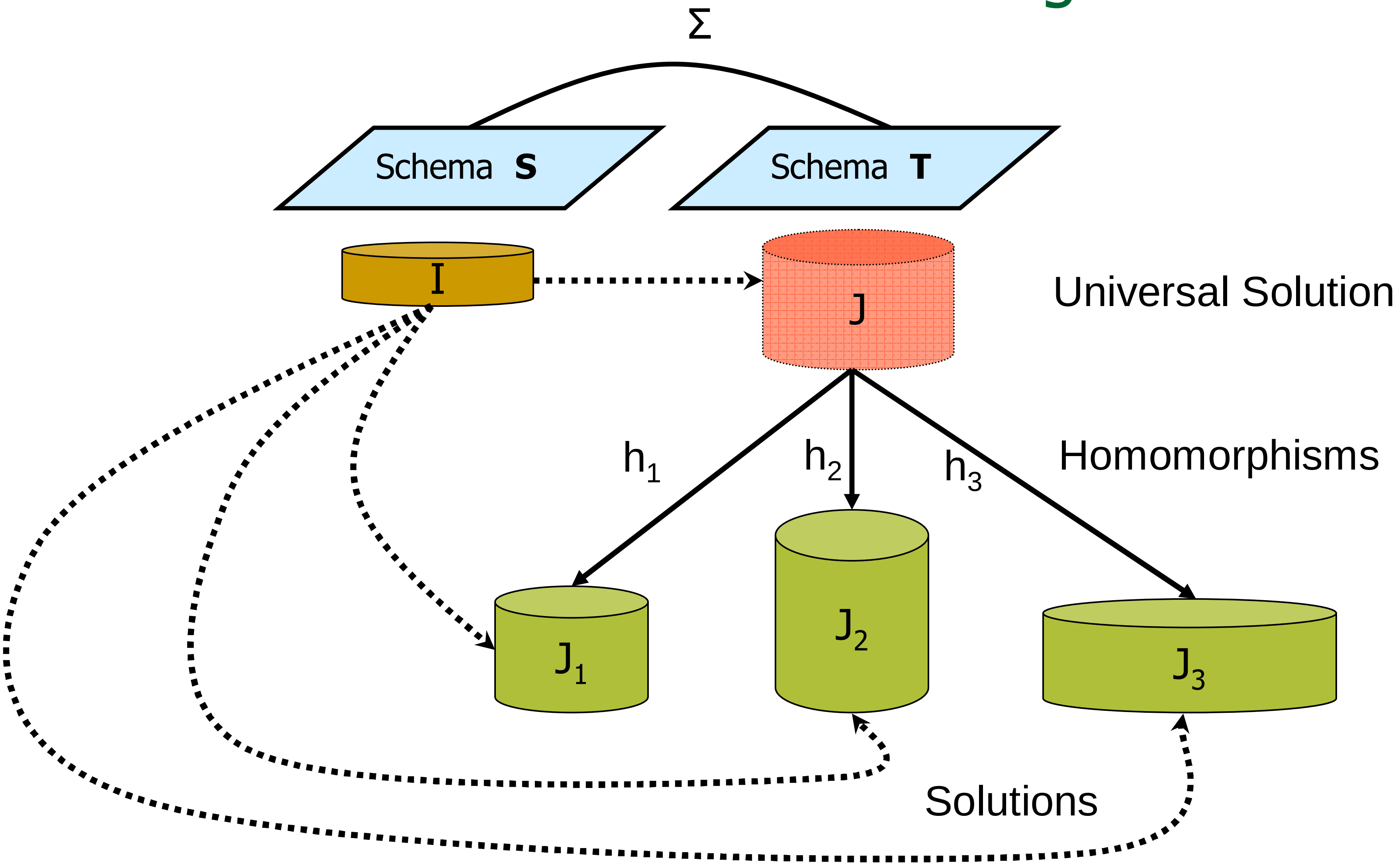
Universal Solutions in Data Exchange

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

- Constants: entries in source instances
- Variables (labeled nulls): other entries in target instances
- Homomorphism h: $J_1 \rightarrow J_2$ between target instances:
  - h(c) = c, for constant c
  - If $P(a_1,\ldots,a_m)$ is in $J_{1,}$, then $P(h(a_1),\ldots,h(a_m))$ is in $J_{2.}$

Claim: Universal solutions are the *preferred* solutions in data exchange.

Universal Solutions in Data Exchange



Example - continued

Source relation S(A,B), target relation T(A,B)

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

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

Solutions: Infinitely many solutions exist

- J_1 = {H(a,b), H(b,b)} is not universal
- J_2 = {H(a,a), H(a,b)} is not universal
- J_3 = {H(a,X), H(X,b)} is universal
- J_4 = {H(a,X), H(X,b), H(a,Y), H(Y,b)} is universal
- J_5 = {H(a,X), H(X,b), 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.

Algorithmic Problems in Data Exchange

Question: What can we say about the complexity of

- The existence-of-solutions problem **Sol(M)**
  and
- The data exchange problem (construct a universal solution)

for a fixed schema mapping **M** = (**S**, **T**, Σ_{st},Σ_t) specified by s-t tgds and target tgds and egds?

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)

Algorithmic Problems in Data Exchange

Proposition: If **M** = (**S**, **T**, Σ_{st}) is a schema mapping such that Σ_{st} is a set of s-t tgds (i.e., no target dependencies), then:

- Solutions always exist; hence, **Sol(M)** is trivial.
- For every source instance I, a universal solution J can be constructed in PTIME using the naïve chase procedure.

Naïve Chase Procedure for **M** = (**S**, **T**, Σ_{st}) : given a source instance I, build a target instance J* that satisfies each s-t tgd in Σ_{st}

- by introducing new facts in J* as dictated by the RHS of the s-t tgd

and

- by introducing new values (variables) in J* each time existential quantifiers need witnesses.

Naïve Chase Procedure

Example: Expanding edges to paths of length 2

Σ_{st}: $E(x,y) \rightarrow \exists z(H(x,z) \wedge H(z,y))$

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

- If E = {(1,2),(2,3)}, then H* = {(1,M),(M,2),(2,N),(N,3)}

  Universal solution for E

Example : Collapsing paths of length 2 to edges

Σ_{st}: $E(x,z) \wedge E(z,y) \rightarrow F(x,y)$

- If E = {(1,3}, (2,4), (3,4)}, then F* = {F(1,4)}

  Universal Solution for E

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

Weakly acyclic sets of tgds contain as special cases:

- **Sets of full tgds (GAV constraints)**

  $$\varphi_T(\mathbf{x},\mathbf{x'}) \rightarrow \psi_T(\mathbf{x}),$$

  where $\varphi_T(\mathbf{x},\mathbf{x'})$ and $\psi_T(\mathbf{x})$ are conjunctions of target atoms.

- **Acyclic sets of inclusion dependencies**

  Large class of dependencies occurring in practice.

Weakly Acyclic Sets of Tgds: Definition

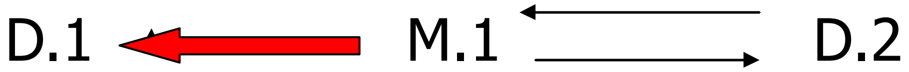
- **Position graph** of a set Σ of tgds:
  - **Nodes:** R.A, with R relation symbol, A attribute of R
  - **Edges:** for every $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ in Σ, for every x in $\mathbf{x}$ occurring in ψ, for every occurrence of x in φ in R.A:
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

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

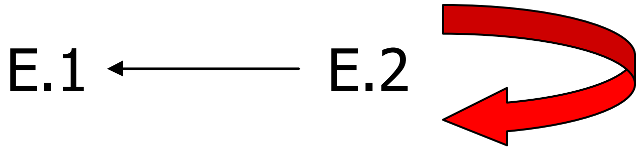
- A tgd θ is **weakly acyclic** if so is the singleton set $\{\theta\}$.

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.



Weak Acyclicity and Chase Termination

Note: If the set of target tgds is **not** weakly acyclic, then the chase procedure may **never** terminate.

Example: $E(x,y) \rightarrow \exists\, z\ E(y,z)$ is not weakly acyclic

$E(1,2) \Rightarrow$

$E(2,X_1) \Rightarrow$

$E(X_1,X_2) \Rightarrow$

$E(X_2, X_3) \Rightarrow$

…

infinite chase

Complexity of Data Exchange

M = (**S**, **T**, Σ_{st} , Σ_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

The Smallest Universal Solution

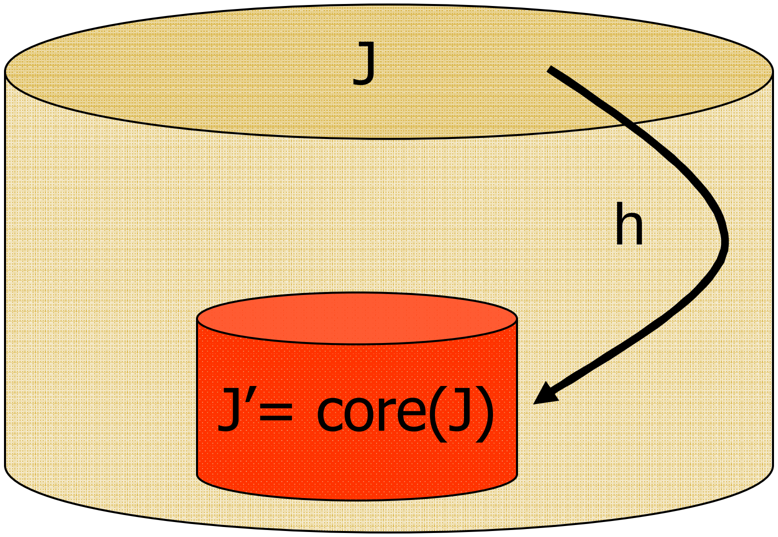
- **Fact:** Universal solutions need not be unique.
- **Question**: Is there a "best" universal solution?
- **Answer:** In joint work with R. Fagin and L. Popa, we took a "small is beautiful" approach:

  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 database has a core.
  - The core is unique up to isomorphism.

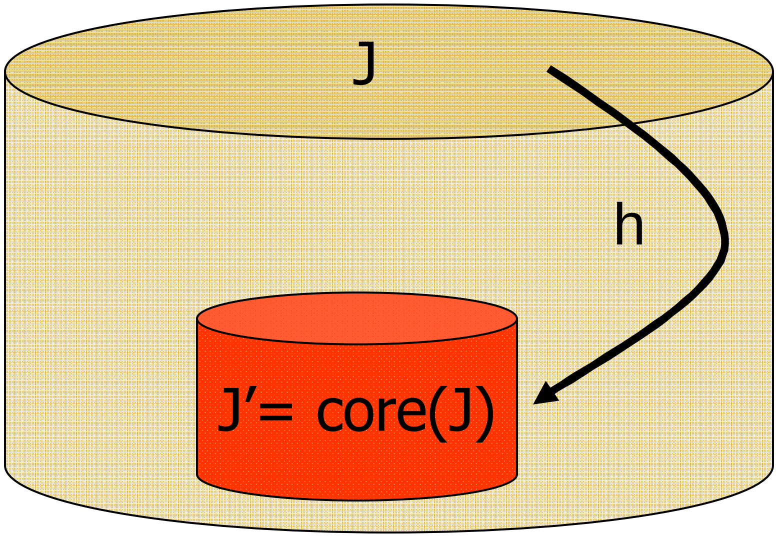
The Core of a Structure



Definition: J′ is the core of J if

- J′ ⊆ J
- there is a hom. h: J → J′
- there is **no** hom. g: J → J″, where J″ ⊂ 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): **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 (Gottlob & Nash): 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.

From Theory to Practice

- Clio Project at IBM Almaden managed by Howard Ho.
    - Semi-automatic schema-mapping generation tool;
    - Data exchange system based on schema mappings.

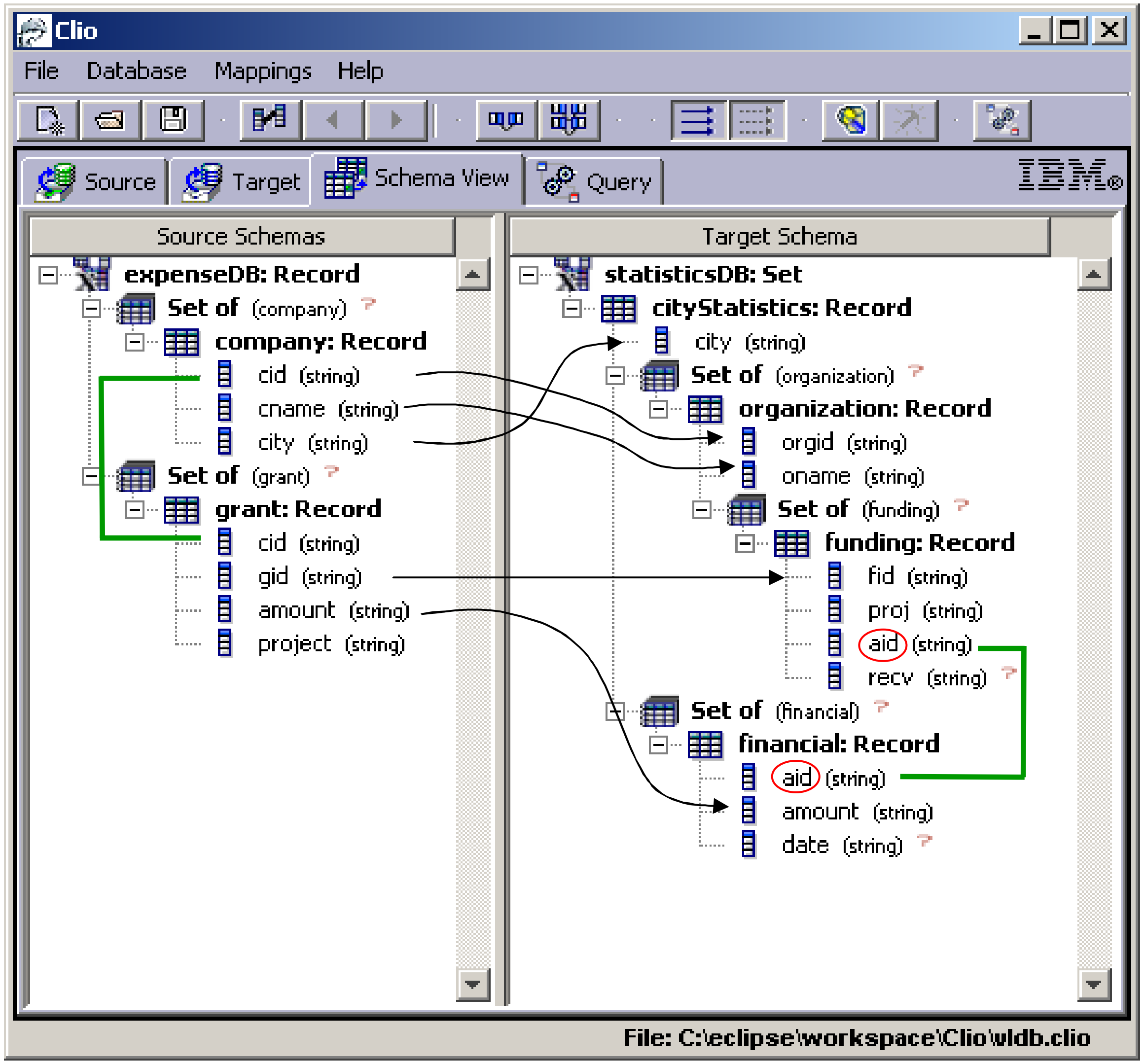
- Universal solutions used as the semantics of data exchange.

- Universal solutions are generated via SQL queries extended with Skolem functions (implementation of chase procedure), provided there are no target constraints.

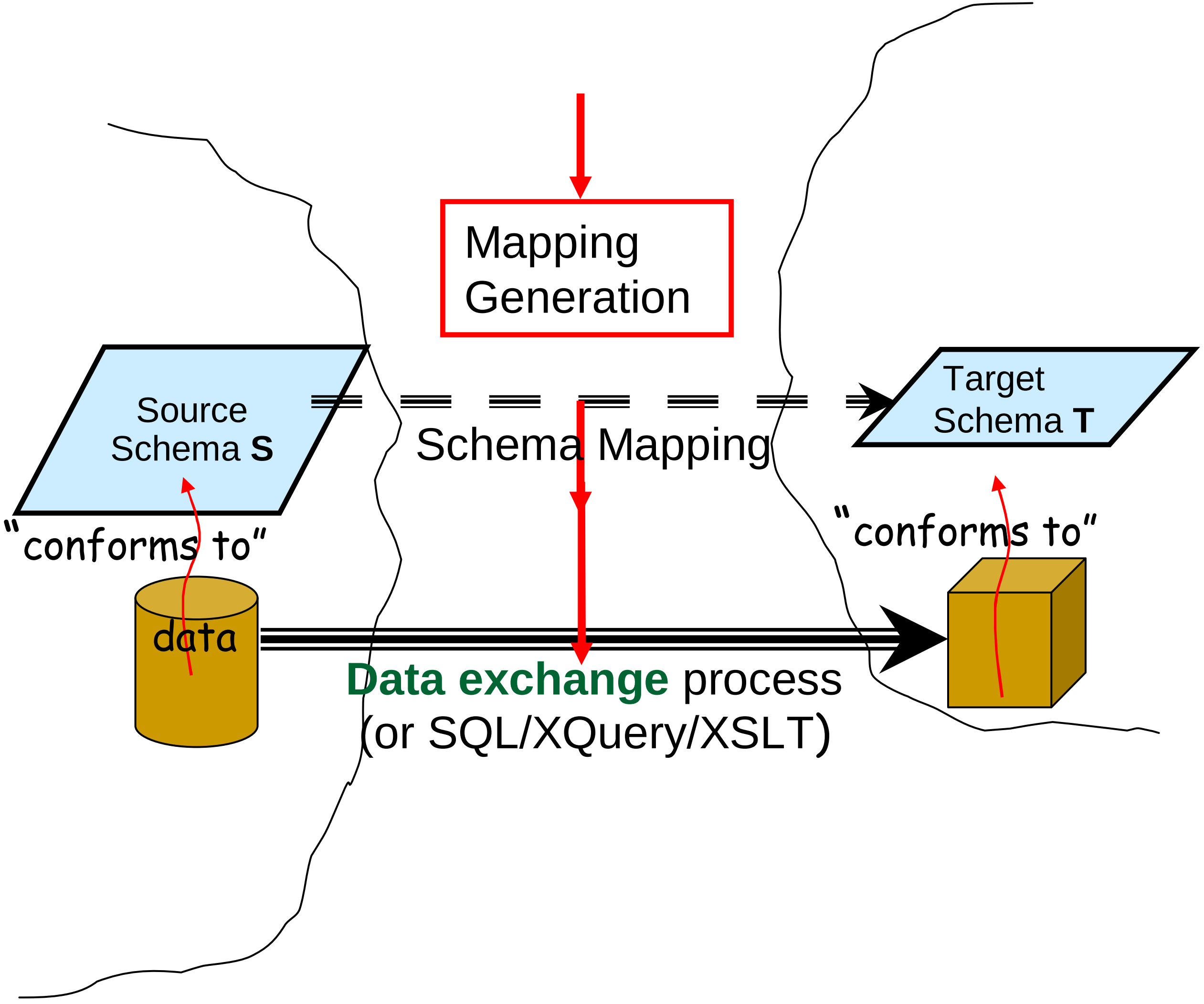
- Clio technology is now part of IBM Rational® Data Architect.

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



Schema Mappings in Clio



Outline

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

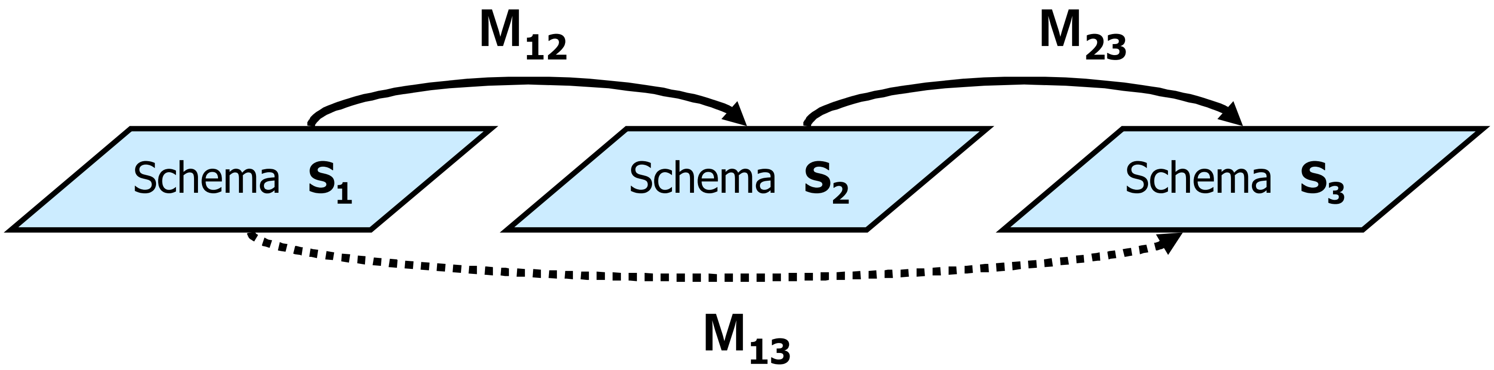
✓ Schema Mappings and Data Exchange
  - ☐ Algorithmic problems in data exchange.
  - ☐ Solutions, universal solutions, and the core.

- 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 generic schema-mapping operators:
  - Match operator
  - Merge operator
  - …
  - Composition operator
  - Inverse operator

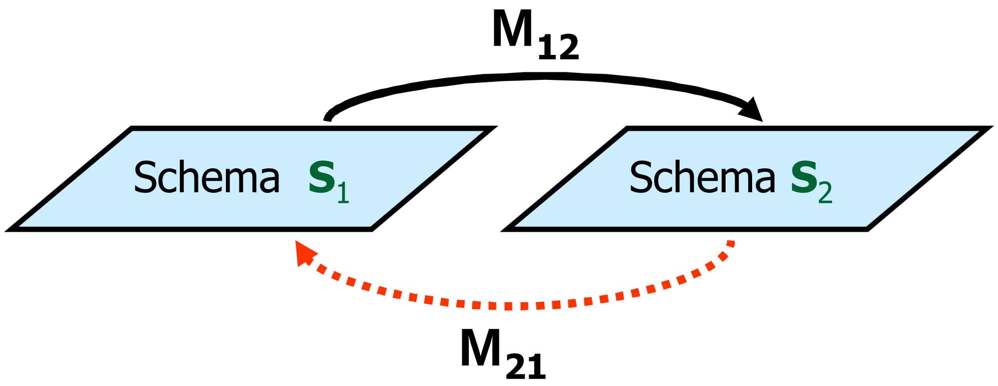
Composing Schema Mappings



- Given $\mathbf{M_{12}} = (\mathbf{S_1}, \mathbf{S_2}, \Sigma_{12})$ and $\mathbf{M_{23}} = (\mathbf{S_2}, \mathbf{S_3}, \Sigma_{23})$, derive a schema mapping $\mathbf{M_{13}} = (\mathbf{S_1}, \mathbf{S_3}, \Sigma_{13})$ that is "equivalent" to the sequential application of $\mathbf{M_{12}}$ and $\mathbf{M_{23}}$.
- $\mathbf{M_{13}}$ is a composition of $\mathbf{M}_{12}$ and $\mathbf{M_{23}}$

$$\mathbf{M_{13}} = \mathbf{M}_{12} \circ \mathbf{M_{23}}$$

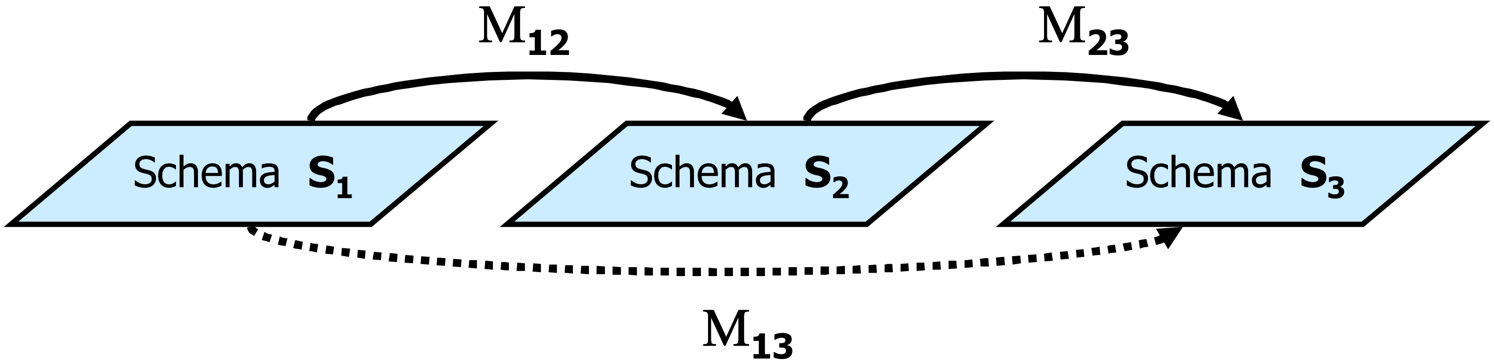
Inverting Schema Mapping



- Given $\mathbf{M}_{12}$, derive $\mathbf{M}_{21}$ that “undoes” $\mathbf{M}_{12}$

$\mathbf{M}_{21}$ is an inverse of $\mathbf{M}_{12}$

Composing Schema Mappings



- Given M_{12} = (**S**$_1$, **S**$_2$, Σ_{12}) and M_{23} = (**S**$_2$, **S**$_3$, Σ_{23}), derive a schema mapping M_{13} = (**S**$_1$, **S**$_3$, Σ_{13}) that is "equivalent" to the sequence M_{12} and M_{23}.

What does it mean for M_{13} to be "equivalent" to the composition of M_{12} and M_{23}?

Earlier Work

- Metadata Model Management (Bernstein in CIDR 2003)
  - Composition is one of the fundamental operators
  - However, no precise semantics is given
- Composing Mappings among Data Sources (Madhavan & Halevy in VLDB 2003)
  - First to propose a semantics for composition
  - However, their definition is in terms of maintaining the same certain answers relative to a class of queries.
  - Their notion of composition *depends* on the class of queries; it may *not* be unique up to logical equivalence.

Semantics of Composition

- Every schema mapping $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ defines a binary relationship Inst($\mathbf{M}$) between instances:

$$\text{Inst}(\mathbf{M}) = \{ (I,J) \mid (I,J) \models \Sigma \}.$$

- **Definition:** (FKPT)

A schema mapping $\mathbf{M}_{13}$ is a composition of $\mathbf{M}_{12}$ and $\mathbf{M}_{23}$ if

$$\text{Inst}(\mathbf{M}_{13}) = \text{Inst}(\mathbf{M}_{12}) \circ \text{Inst}(\mathbf{M}_{23}), \text{ that is,}$$

$$(I_1,I_3) \models \Sigma_{13}$$

if and only if

there exists I_2 such that $(I_1,I_2) \models \Sigma_{12}$ and $(I_2,I_3) \models \Sigma_{23}$.

- **Note:** Also considered by S. Melnik in his Ph.D. thesis

The Composition of Schema Mappings

Fact: If both $\mathrm{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma)$ and $\mathrm{M}' = (\mathbf{S}_1, \mathbf{S}_3, \Sigma')$ are compositions of M_{12} and M_{23}, then Σ are Σ' are logically equivalent. For this reason:

- We say that M (or M') is *the* composition of M_{12} and M_{23}.
- We write $\mathrm{M}_{12} \circ \mathrm{M}_{23}$ to denote it

Issues in Composition of Schema Mappings

- The semantics of composition was the first main issue.

Some other key issues:

- Is the language of s-t tgds *closed under composition*?

  If M_{12} and M_{23} are specified by finite sets of s-t tgds, is $M_{12} \circ M_{23}$ also specified by a finite set of s-t tgds?

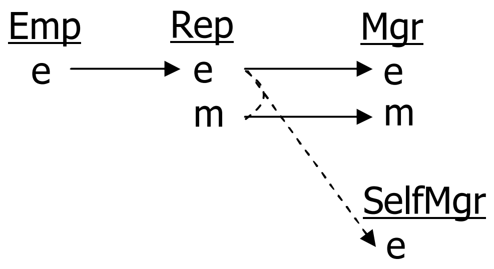
- If not, what is the "right" language for composing schema mappings?

Composition: Expressibility

M_{12} Σ_{12}	M_{23} Σ_{23}	$M_{12} \circ M_{23}$ Σ_{13}
finite set of GAV (full) s-t tgds $\varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x})$	finite set of s-t tgds $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y})$	finite set of s-t tgds $\varphi(\mathbf{x})\rightarrow\exists \mathbf{y}\psi(\mathbf{x},\mathbf{y})$
finite set of s-t tgds $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x},\mathbf{y})$	finite set of s-t tgds $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y})$	may not be definable: by any set of s-t tgds; in FO-logic; in Datalog.

Employee Example

- Σ_{12} :
  - Emp(e) → ∃m Rep(e,m)
- Σ_{23} :
  - Rep(e,m) → Mgr(e,m)
  - Rep(e,e) → SelfMgr(e)



- **Theorem:** This composition is not definable by any finite set of s-t tgds.

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

Employee Example - revisited

Σ_{12} :

- $\forall e\ (\ Emp(e) \rightarrow \exists m\ Rep(e,m)\ )$

Σ_{23} :

- $\forall e \forall m(\ Rep(e,m) \rightarrow Mgr(e,m)\ )$
- $\forall e\ (\ Rep(e,e) \rightarrow SelfMgr(e)\ )$

Fact: The composition is definable by the SO-tgd

Σ_{13} :

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

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

Composing SO-Tgds and Data Exchange

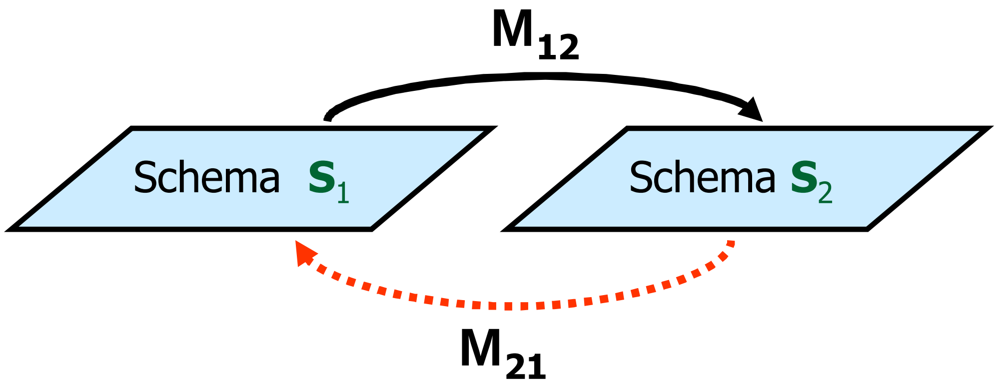
Theorem (FKPT):

- 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 finite sets of s-t tgds. Hence, SO tgds are the "right" language for the composition of s-t tgds

Synopsis of Schema Mapping Composition

- s-t tgds are not closed under composition.
- SO-tgds form a well-behaved fragment of second-order logic.
  - SO-tgds are closed under composition; they are the "right" language for composing s-t tgds.
  - SO-tgds are "chasable": Polynomial-time data exchange with universal solutions.
- SO-tgds and the composition algorithm have been incorporated in Clio's Mapping Specification Language (MSL).

Inverting Schema Mapping



- Given $\mathbf{M_{12}}$, derive $\mathbf{M_{21}}$ that "undoes" $\mathbf{M_{12}}$

$\mathbf{M_{21}}$ is an inverse of $\mathbf{M_{12}}$

- What is the "right" semantics of the inverse operator?

Inverting Schema Mappings

In recent years, three different approaches to inverting schema mappings have been proposed and investigated:

- A notion of inverse introduced by Fagin in 2006;
- A notion of quasi-inverse introduced by Fagin, K …, Popa, and Tan in 2007.
- A notion of maximum recovery introduced by Arenas, Pérez, and Riveros in 2008.

Thus far, no definitive notion of the inverse operator has emerged.
So the research goes on …

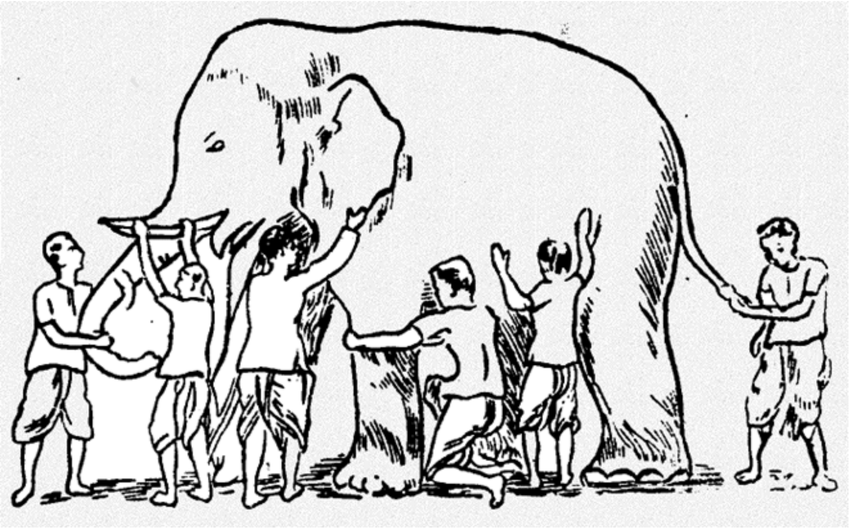
Some Directions of Research

- Inverting schema mappings requires further study.
- Detailed study of other schema mapping operators (Diff, Merge, …) remains to be carried out.
- Applications of schema-mapping operators to:
  - Study of schema evolution;
  - Modeling and analysis of ETL via schema mappings.

Related Work (very partial list)

- XML Data Exchange
  (Arenas and Libkin – 2005).
- Schema mappings with arithmetic comparisons
  (Afrati, Li, Pavlaki – 2008).
- Composing richer schema mappings
  (Nash, Bernstein, Melnik – 2007)
- Peer data exchange
  (Fuxman, K …, Miller, Tan – 2007)
- Schema-mapping optimization
  (FKNP – 2008)

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.