Schema Mappings
Data Exchange
&
Metadata Management

Phokion G. Kolaitis
IBM Almaden Research Center

joint work with

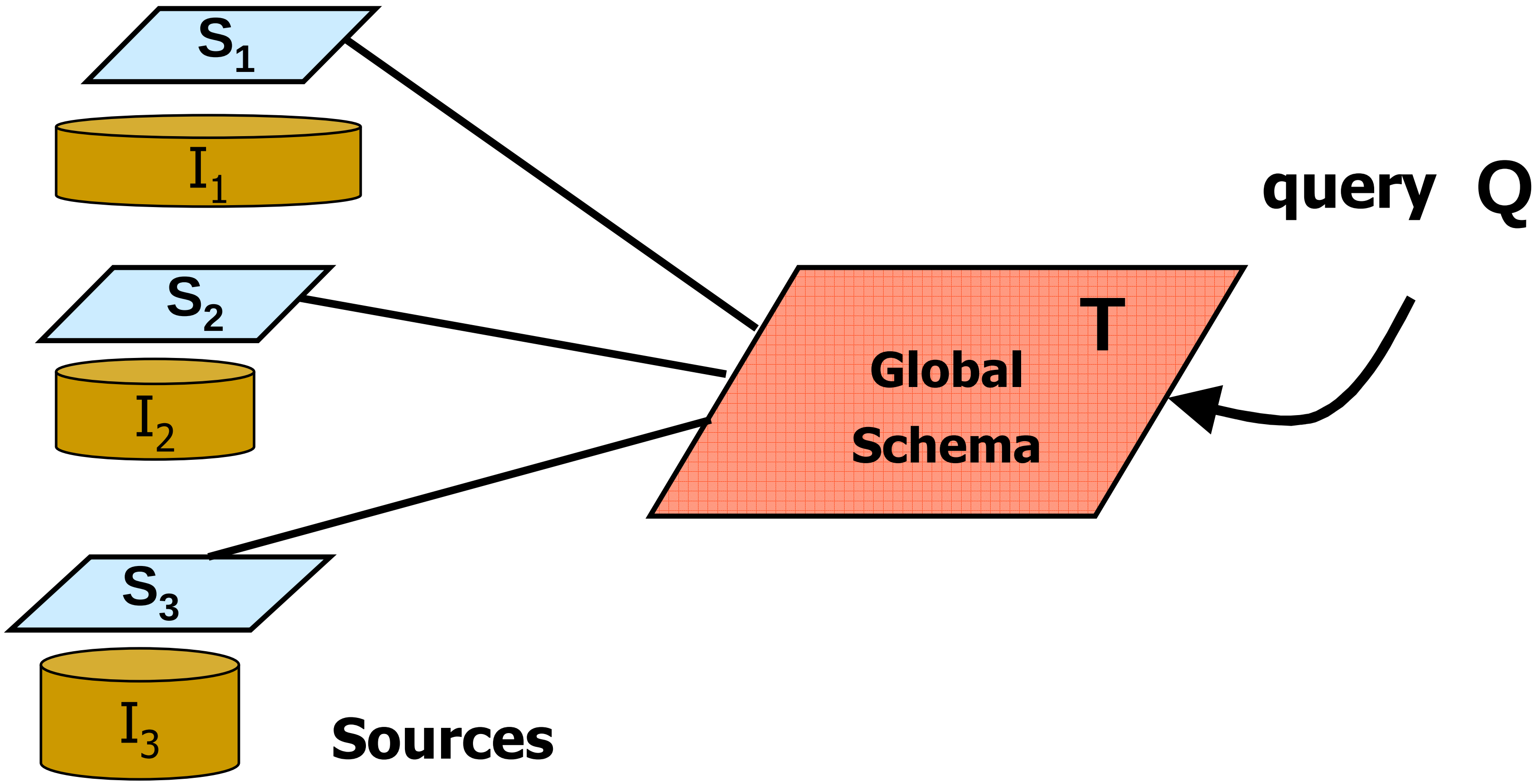
Ronald Fagin	Renée J. Miller	Lucian Popa	Wang-Chiew Tan
IBM Almaden	U. Toronto	IBM Almaden	UC Santa Cruz

The Data Interoperability Problem

- Data may reside
  - at several different sites
  - in several different formats (relational, XML, …).
- Two different, but related, facets of data interoperability:
  - **Data Integration** (aka **Data Federation**):
  - **Data Exchange** (aka **Data Translation**):

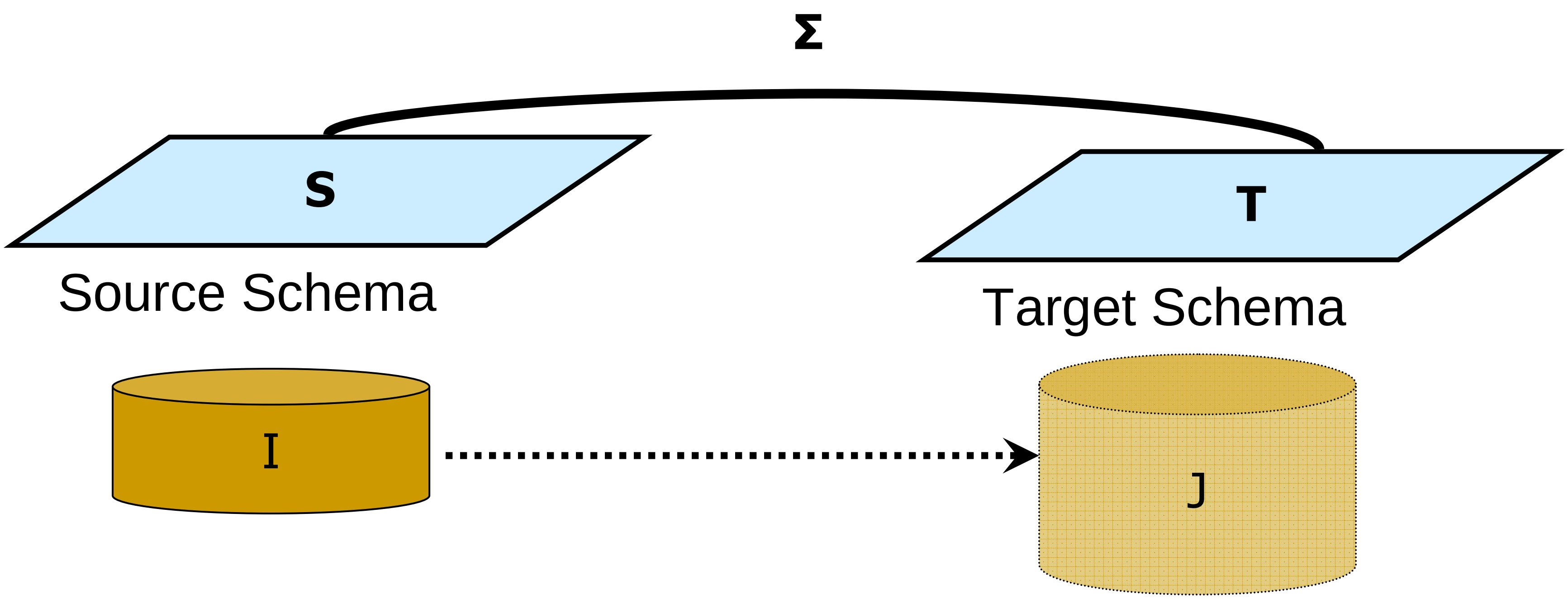
Data Integration

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.



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:
  - Data Warehousing, ETL (Extract-Transform-Load) tasks;
  - XML Publishing, XML Storage, …

Foundations of Data Interoperability

Theoretical Aspects of Data Interoperability

Develop a conceptual framework for formulating and studying fundamental problems in data interoperability:

- Semantics of data integration & data exchange
- Algorithms for data exchange
- Complexity of query answering

Outline of the Talk

- Schema Mappings and Data Exchange
- Solutions in Data Exchange
  - Universal Solutions
  - The Core of the Universal Solutions
- Query Answering in Data Exchange
- Composing Schema Mappings

Schema Mappings

- Schema mappings:

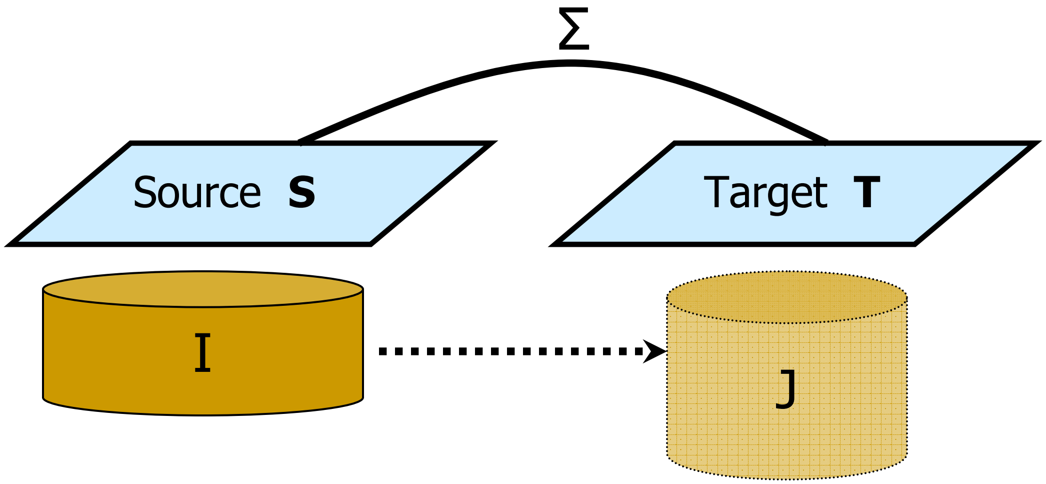
  high-level, declarative assertions that specify the relationship between two schemas.

- Ideally, schema mappings should be
  - expressive enough to specify data interoperability tasks;
  - simple enough to be efficiently manipulated by tools.

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

- Schema mappings play a prominent role in Bernstein's metadata management framework.

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**.

Solutions in Schema Mappings

Definition: Schema Mapping **M** = (**S**, **T**, Σ)

If I is a source instance, then a solution for I is a target instance J such that <I, J > satisfy Σ.

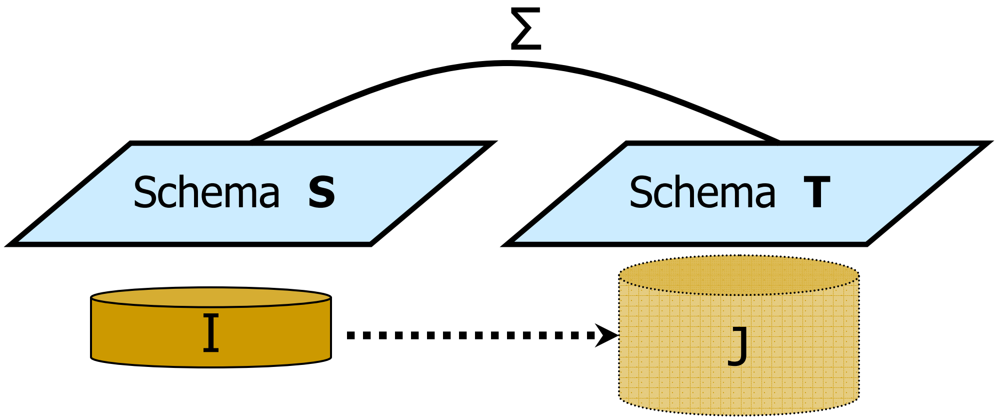
Fact: In general, for a given source instance I,

- No solution for I may exist

  or

- Multiple solutions for I may exist; in fact, infinitely many solutions for I may exist.

Schema Mappings: Basic Problems



Definition: Schema Mapping **M** = (**S**, **T**, Σ)

- 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.

Schema Mapping Specification Languages

- **Question**: How are schema mappings specified?
- **Answer**: Use logic. In particular, it is natural to try to use first-order logic as a specification language for schema mappings.
- **Fact**: There is a fixed first-order sentence specifying a schema mapping **M*** such that **Sol(M*)** is undecidable.
- Hence, we need to restrict ourselves to well-behaved fragments of first-order logic.

Embedded Implicational Dependencies

- Dependency Theory: extensive study of constraints in relational databases in the 1970s and 1980s.

- Embedded Implicational Dependencies: Fagin, Beeri-Vardi, …
  Class of constraints with a balance between high expressive power and good algorithmic properties:
  - Tuple-generating dependencies (tgds)

  Inclusion and multi-valued dependencies are a special case.
  - Equality-generating dependencies (egds)

  Functional dependencies are a special case.

Data Exchange with Tgds and Egds

- Joint work with R. Fagin, R.J. Miller, and L. Popa
- Studied data exchange between relational schemas for schema mappings specified by
  - Source-to-target tgds
  - Target tgds
  - Target egds

Schema Mapping Specification Language

The relationship between source and target is given by formulas of first-order logic, called

Source-to-Target Tuple Generating Dependencies (s-t tgds)

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

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

Example:

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

Schema Mapping Specification Language

- s-t tgds assert that:
  some SPJ source query is *contained* in some other SPJ target query

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

- s-t tgds generalize the main specifications used in data integration:
  - They generalize LAV (local-as-view) specifications:

    $P(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$, where P is a source schema.
  - They generalize GAV (global-as-view) specifications:

    $\varphi(\mathbf{x}) \rightarrow R(\mathbf{x})$, where R is a target schema
  - At present, most commercial II systems support GAV only.

Target Dependencies

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

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

  Dept (did, dname, mgr_id, mgr_name) $\rightarrow$ Mgr (mgr_id, did)

  (a target inclusion dependency constraint)

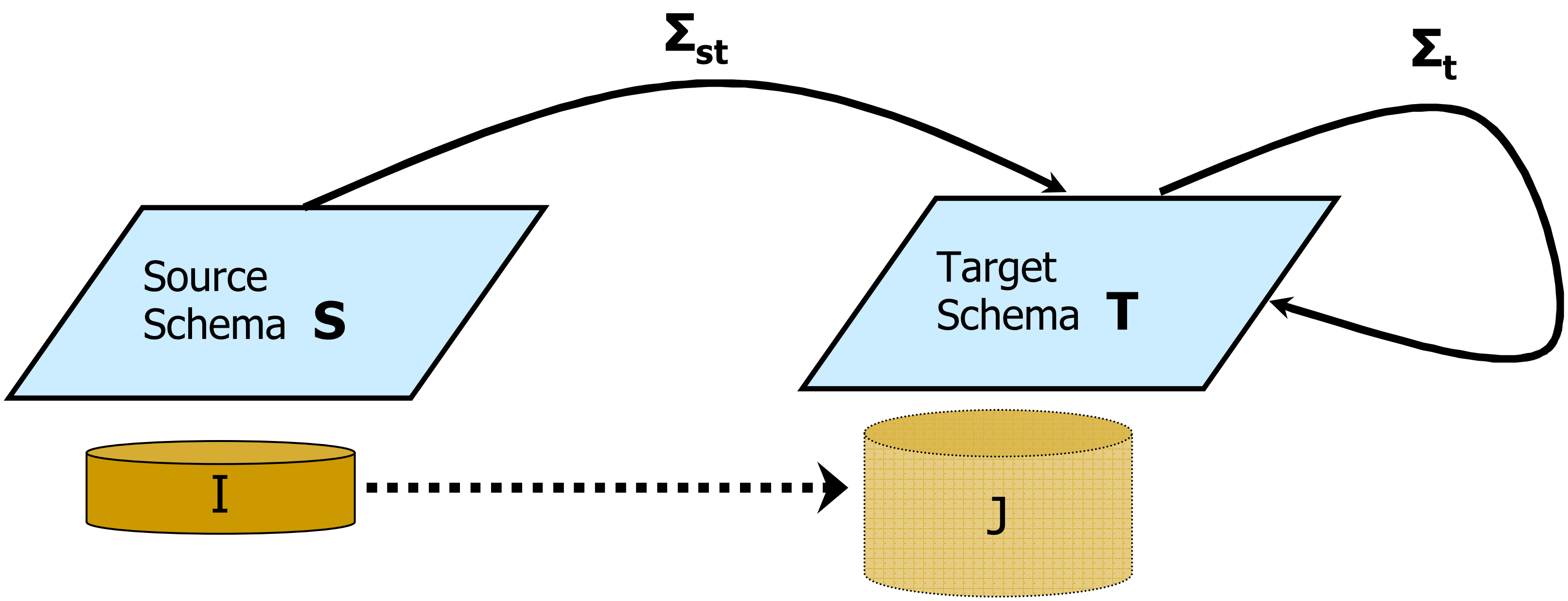
- Target Equality Generating Dependencies (egds):

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

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

  (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

Underspecification in Data Exchange

- **Fact:** Given a source instance, multiple solutions may exist.

- **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

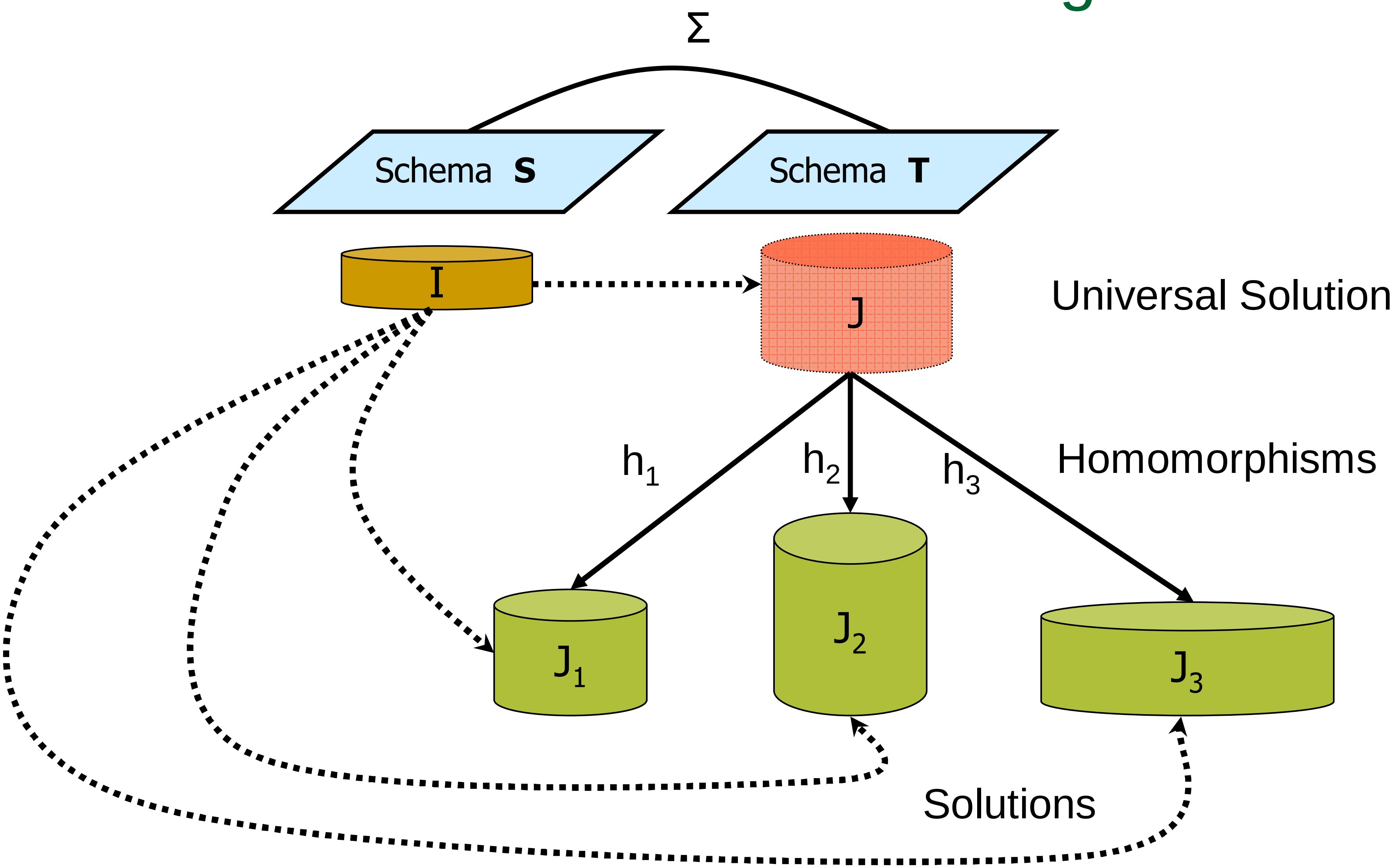
For a given source instance, there may be multiple target instances satisfying the specifications of the schema mapping. Thus,

- 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

- We introduced the notion of universal solutions as the "best" solutions in data exchange.
  - By definition, 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

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 analogous to most general unifiers in logic programming.
- 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 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),** the existence-of-solutions problem for **M**, is in P.
- A *canonical* universal solution (if solutions exist) can be produced in polynomial time using the chase procedure.

Weakly Acyclic Sets of Tgds

Weakly acyclic sets of tgds contain as special cases:

- **Sets of full tgds**

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

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

  Example: $H(x,z) \wedge H(z,y) \rightarrow H(x,y) \wedge C(z)$

  Full tgds express containment between relational joins.

- **Sets of acyclic inclusion dependencies**

  Large class of dependencies occurring in practice.

The Smallest Universal Solution

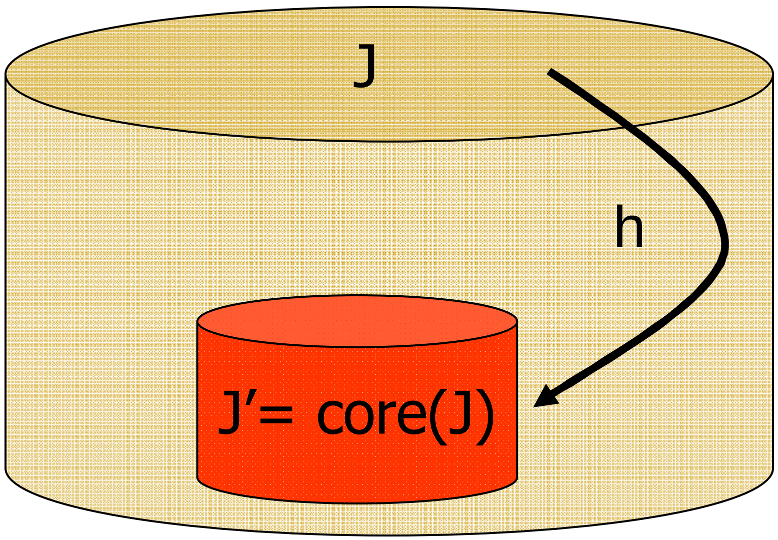
- **Fact:** Universal solutions need not be unique.
- **Question**: Is there a "best" universal solution?
- **Answer:** In joint work with R. Fagin and L. Popa, we took a "small is beautiful" approach:

  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.

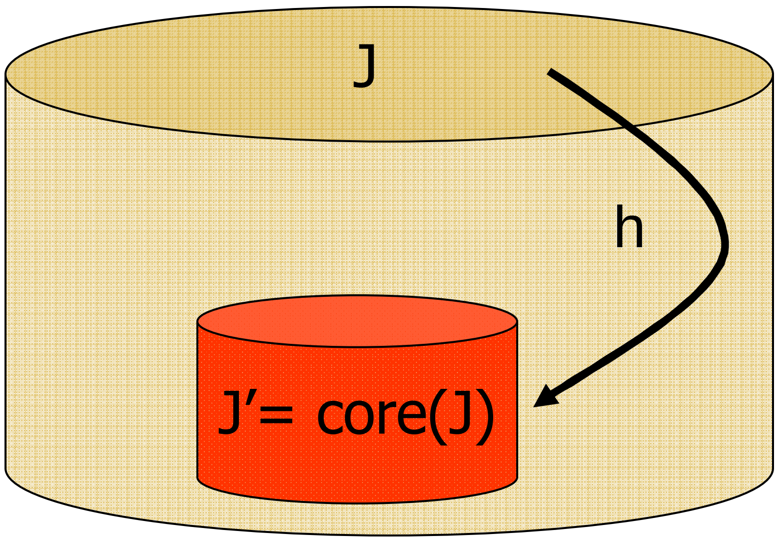
The Core of a Structure



Definition: J′ is the core of J if

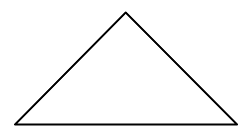
- J′ ⊆ J
- there is a hom. h: J → J′
- there is **no** hom. g: J → J″, where J″ ⊂ J′.

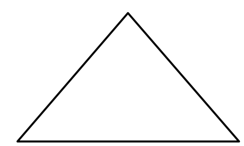
The Core of a Structure



Definition: J' is the core of J if

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

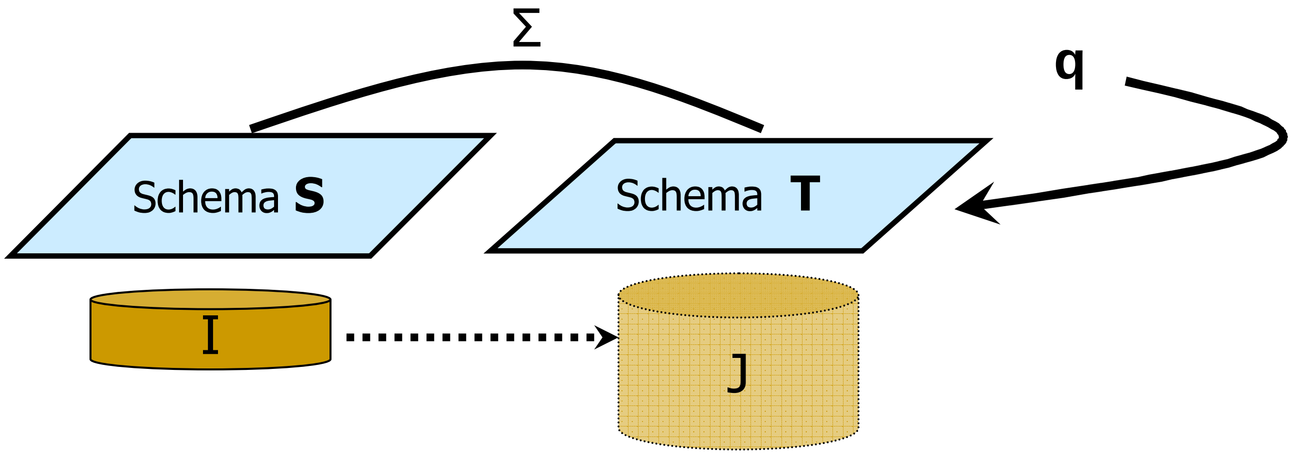
Theorem (Gottlob – PODS 2005): **M** = (**S**, **T**, Σ_{st}, Σ_t)

If every target constraint is an egd or a full tgd, then the core is polynomial-time computable.

Outline of the Talk

- ✓ Schema Mappings and Data Exchange
- ✓ Solutions in Data Exchange
  - ✓ Universal Solutions
  - ✓ The Core of the Universal Solutions
- Query Answering in Data Exchange
- Composing Schema Mappings

Query Answering in Data Exchange



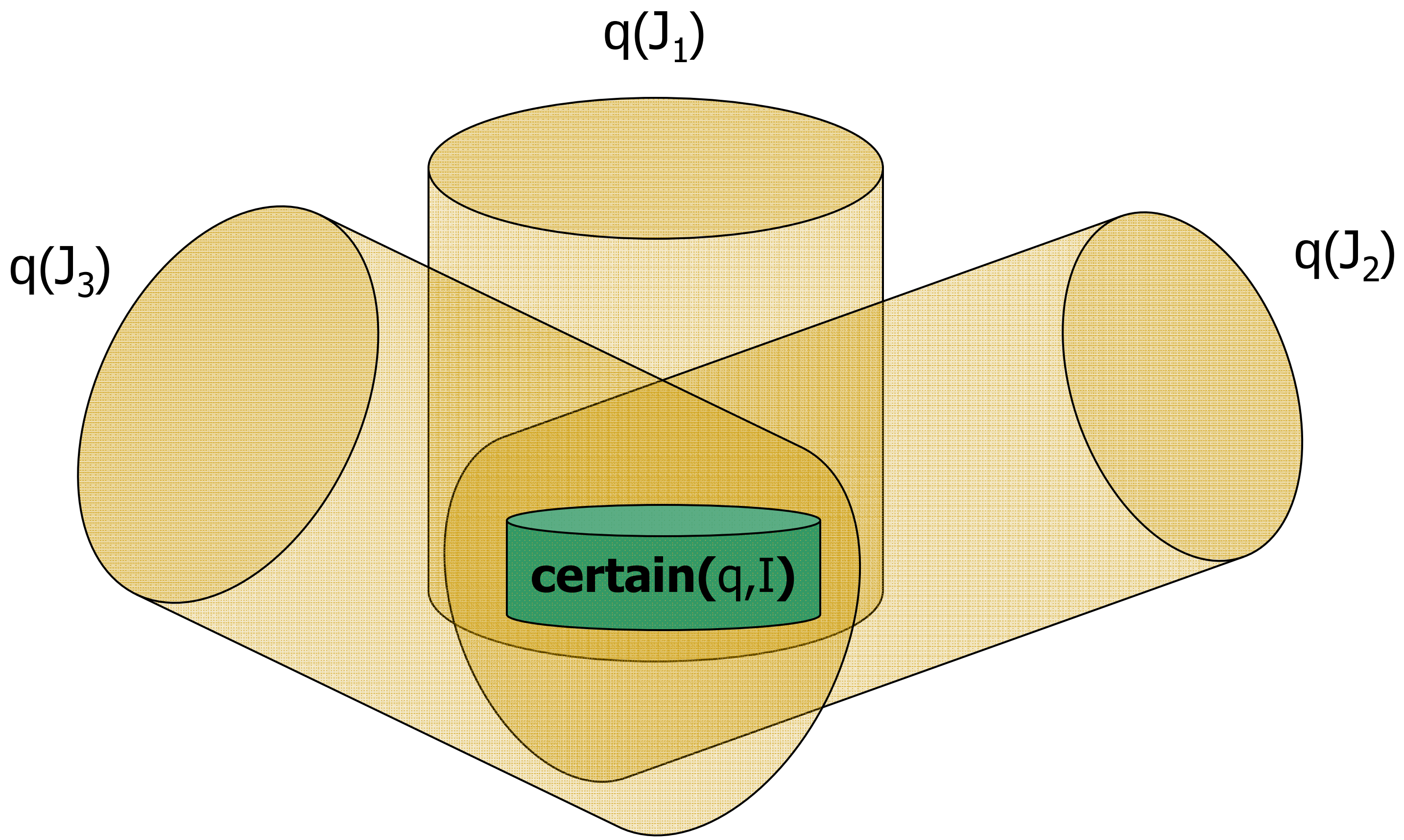
Question: What is the semantics of target query answering?

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 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**.

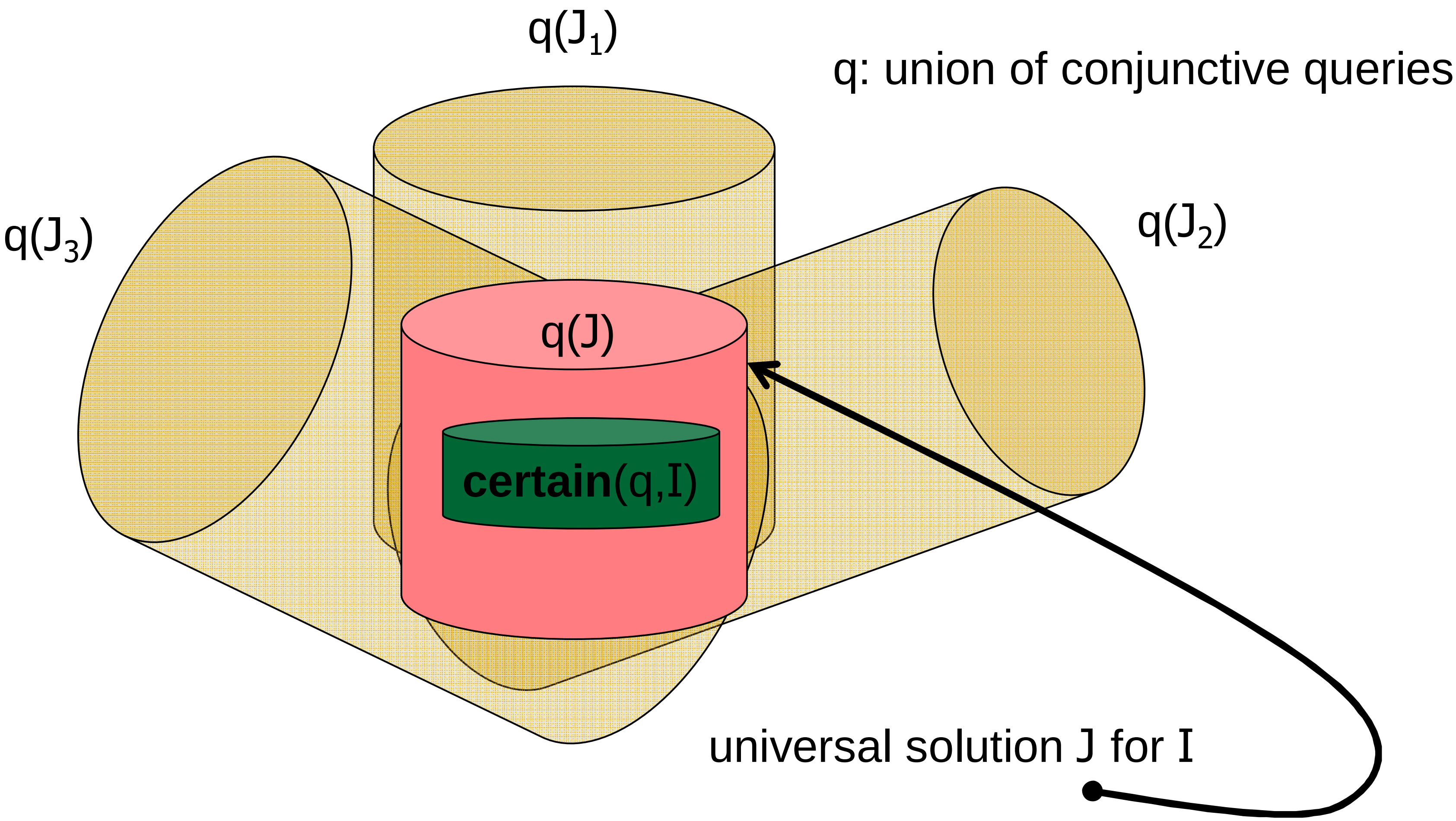
- If I is a source instance and J is a universal solution for I, then

  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 $\Sigma_t = \emptyset$.

Universal Certain Answers

- Alternative semantics of query answering based on universal solutions.
- Certain Answers:

  "Possible Worlds" = Solutions
- Universal Certain Answers:

  "Possible Worlds" = Universal Solutions

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

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

Facts:

- $\textbf{certain}(q,I) \subseteq \textbf{u-certain}(q,I)$
- $\textbf{certain}(q,I) = \textbf{u-certain}(q,I)$, q a union of conjunctive queries

Computing the Universal Certain Answers

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

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

Let q be an existential query over **T**.

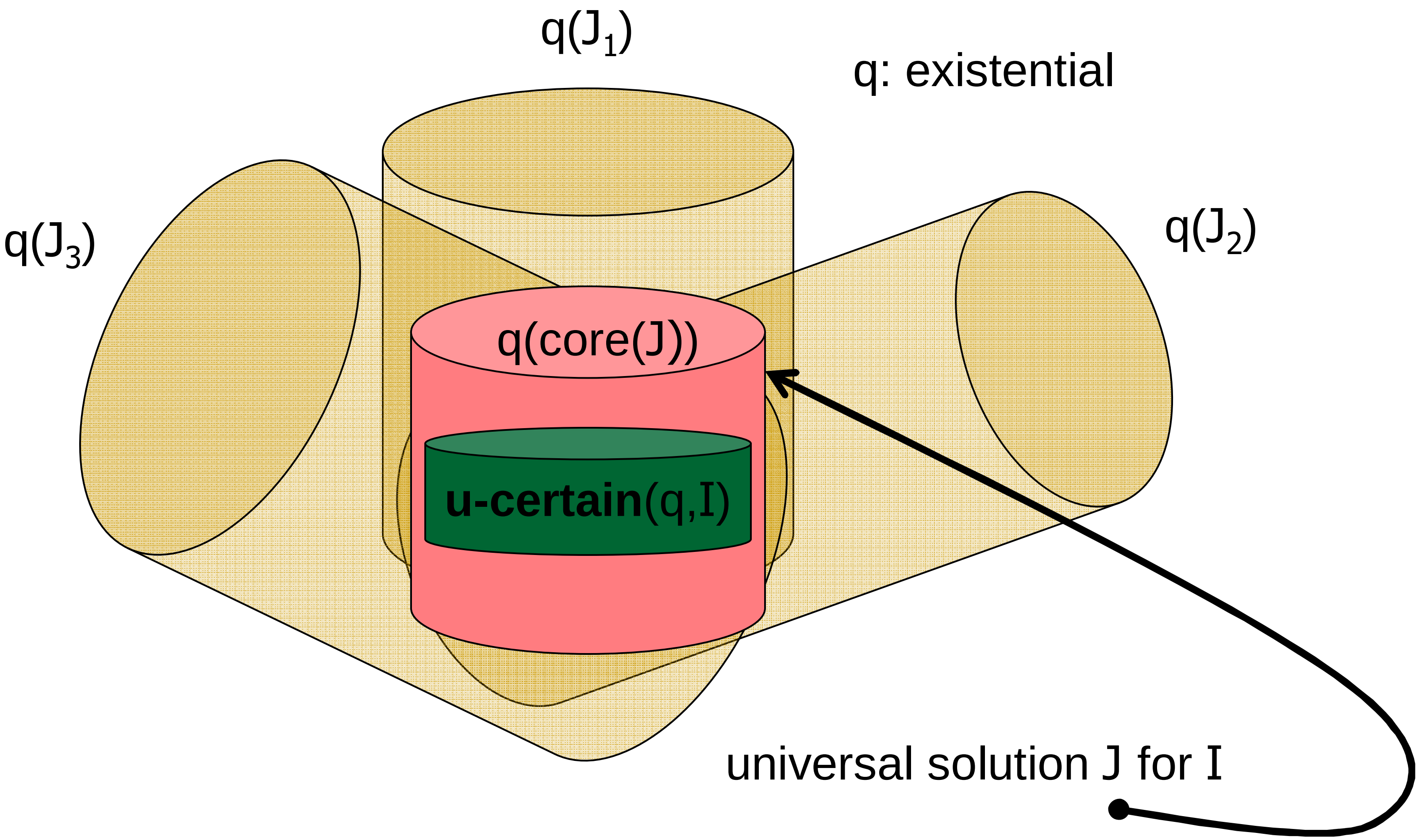
- If I is a source instance and J is a universal solution for I, then

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

- Hence, **u-certain**(q,I) is computable in time polynomial in |I| whenever the core of the universal solutions is polynomial-time computable.

Note: Unions of conjunctive queries with inequalities are a special case of existential queries.

Universal Certain Answers via the Core



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

From Theory to Practice

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

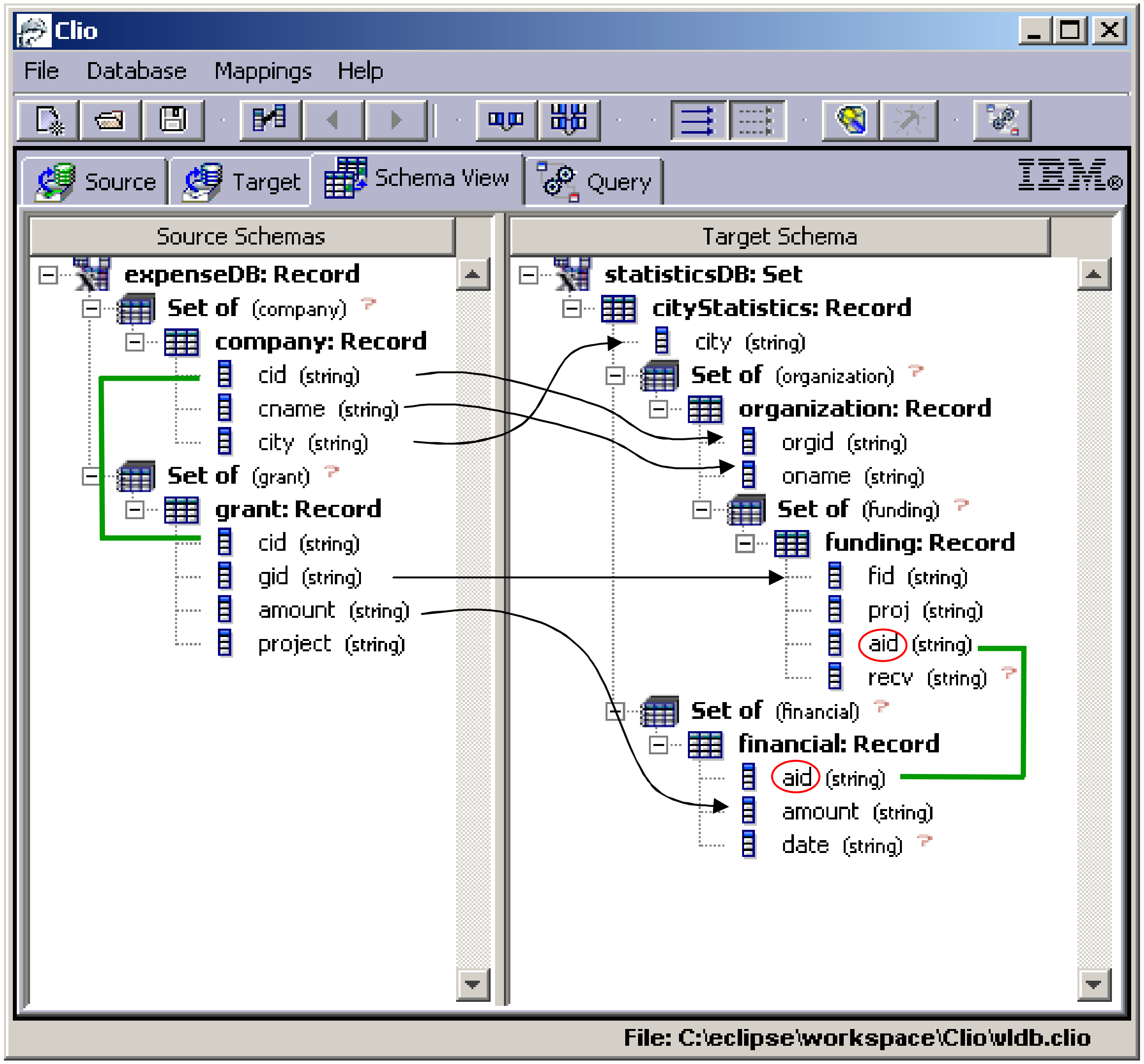
- Universal solutions used as the semantics of data exchange.

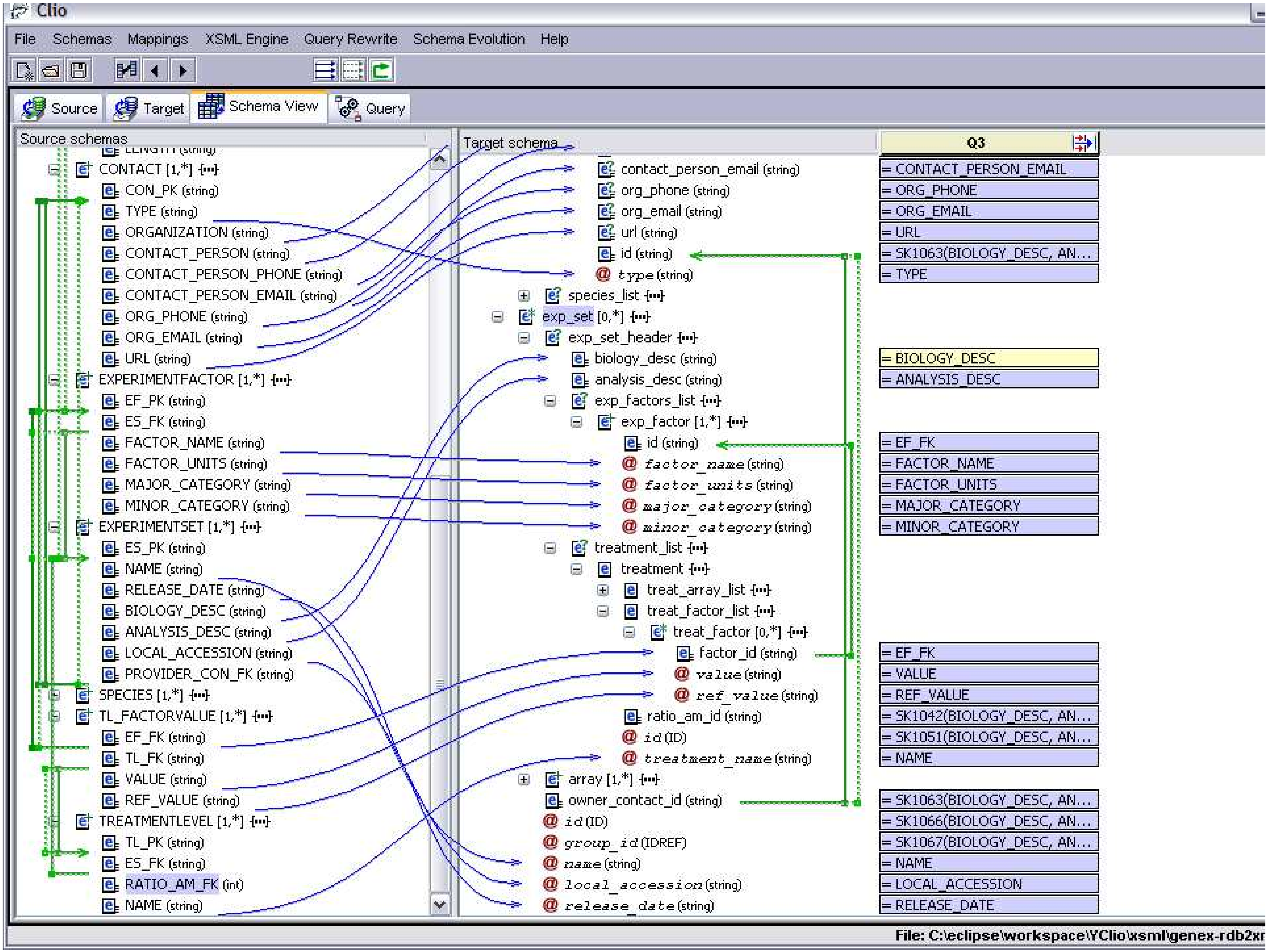
- Universal solutions are generated via SQL queries extended with Skolem functions (implementation of chase procedure), provided there are no target constraints.

- Clio/Criollo technology is being exported to WebSphere II.

Some Features of Clio

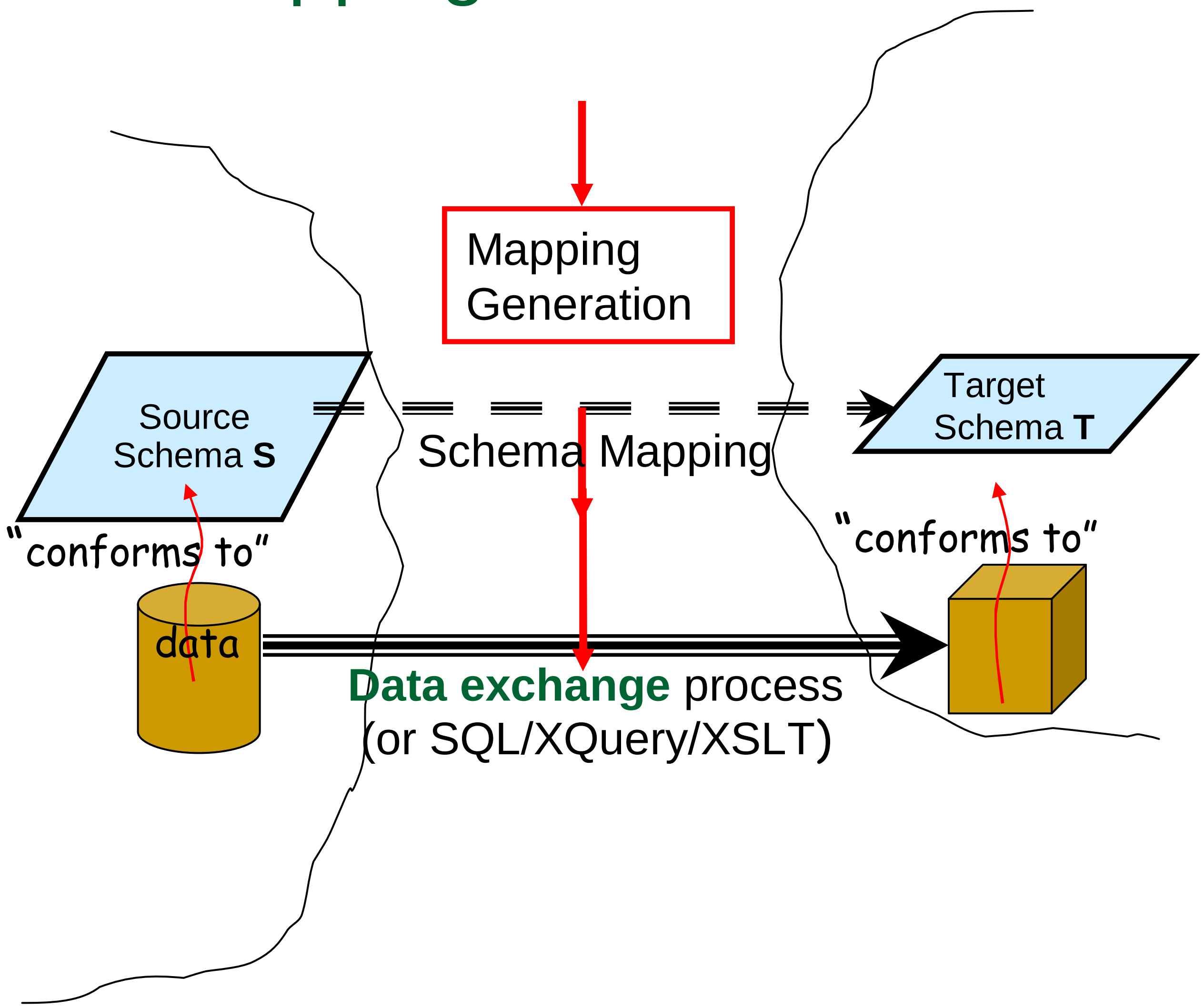
- Supports nested structures
  - Nested Relational Model
  - Nested Constraints
- Automatic & semi-automatic discovery of attribute correspondence.
- Interactive derivation of schema mappings.
- Performs data exchange





Clio
File
Schemas
Mappings
XSML Engine
Query Rewrite
Schema Evolution
Help
Source
Target
Schema View
Query
Source schemas
CONTACT [1,*]
CON_PK (string)
TYPE (string)
ORGANIZATION (string)
CONTACT_PERSON (string)
CONTACT_PERSON_PHONE (string)
CONTACT_PERSON_EMAIL (string)
ORG_PHONE (string)
ORG_EMAIL (string)
URL (string)
EXPERIMENTFACTOR [1,*]
EF_PK (string)
ES_FK (string)
FACTOR_NAME (string)
FACTOR_UNITS (string)
MAJOR_CATEGORY (string)
MINOR_CATEGORY (string)
EXPERIMENTSET [1,*]
ES_PK (string)
NAME (string)
RELEASE_DATE (string)
BIOLOGY_DESC (string)
ANALYSIS_DESC (string)
LOCAL_ACCESSION (string)
PROVIDER_CON_FK (string)
SPECIES [1,*]
TL_FACTORVALUE [1,*]
EF_FK (string)
TL_FK (string)
VALUE (string)
REF_VALUE (string)
TREATMENTLEVEL [1,*]
TL_PK (string)
ES_FK (string)
RATIO_AM_FK (int)
NAME (string)
Target schema
contact_person_email (string)
org_phone (string)
org_email (string)
url (string)
id (string)
@ type (string)
species_list
exp_set [0,*]
exp_set_header
biology_desc (string)
analysis_desc (string)
exp_factors_list
exp_factor [1,*]
id (string)
@ factor_name (string)
@ factor_units (string)
@ major_category (string)
@ minor_category (string)
treatment_list
treatment
treat_array_list
treat_factor_list
treat_factor [0,*]
factor_id (string)
@ value (string)
@ ref_value (string)
ratio_am_id (string)
@ id (ID)
@ treatment_name (string)
array [1,*]
owner_contact_id (string)
@ id (ID)
@ group_id (IDREF)
@ name (string)
@ local_accession (string)
@ release_date (string)
Q3
= CONTACT_PERSON_EMAIL
= ORG_PHONE
= ORG_EMAIL
= URL
= SK1063(BIOLOGY_DESC, AN...
= TYPE
= BIOLOGY_DESC
= ANALYSIS_DESC
= EF_FK
= FACTOR_NAME
= FACTOR_UNITS
= MAJOR_CATEGORY
= MINOR_CATEGORY
= EF_FK
= VALUE
= REF_VALUE
= SK1042(BIOLOGY_DESC, AN...
= SK1051(BIOLOGY_DESC, AN...
= NAME
= SK1063(BIOLOGY_DESC, AN...
= SK1066(BIOLOGY_DESC, AN...
= SK1067(BIOLOGY_DESC, AN...
= NAME
= LOCAL_ACCESSION
= RELEASE_DATE
File: C:\eclipse\workspace\YClio\xsml\genex-rdb2xr

Schema Mappings in Clio



Outline of the Talk

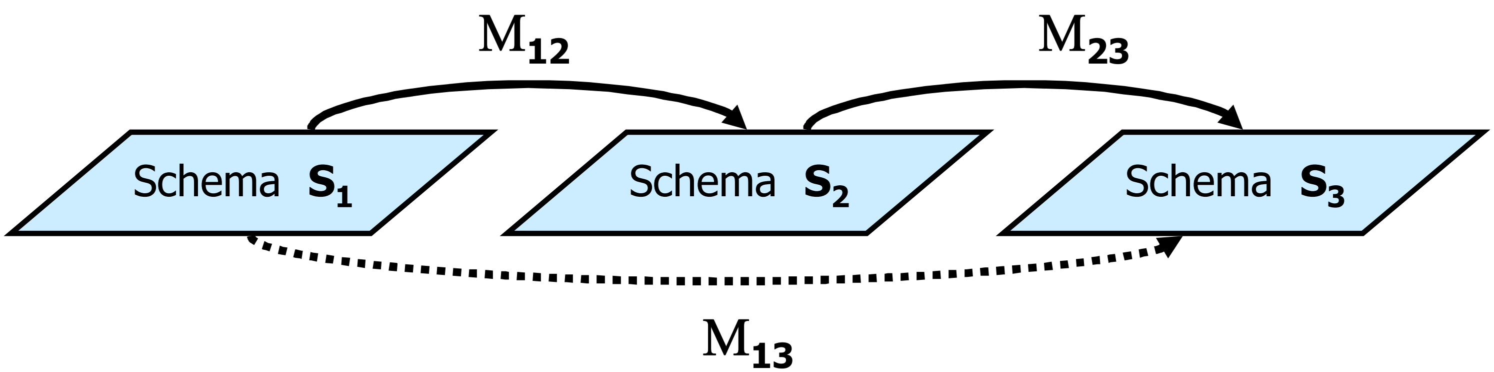
- ✓ Schema Mappings and Data Exchange
- ✓ Solutions in Data Exchange
  - ✓ Universal Solutions
  - ✓ The Core of the Universal Solutions
- ✓ Query Answering in Data Exchange
- Composing Schema Mappings

  joint work with R. Fagin, L. Popa, and W.-C. Tan

Managing Schema Mappings

- Schema mappings can be quite complex.
- Methods and tools are needed to manage schema mappings automatically.
- Metadata Management Framework – Bernstein 2003
  based on generic schema-mapping operators:
  - Composition operator
  - Inverse operator
  - Merge operator
  - ….

Composing Schema Mappings



- Given $M_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $M_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, derive a schema mapping $M_{13} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ 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 **M** = (**S**, **T**, Σ) defines a binary relationship Inst(**M**) between instances*:*

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

- **Definition:** (FKPT)

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

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

  $$\langle I_1, I_3 \rangle \models \Sigma_{13}$$

  if and only if

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

- **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

Definition: The composition query of M_{12} and M_{23} is the set

$$\mathrm{Inst}(\mathrm{M}_{12}) \circ \mathrm{Inst}(\mathrm{M}_{23})$$

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 $\mathrm{M}_{12} \circ \mathrm{M}_{23}$ also specified by a finite set of s-t tgds?

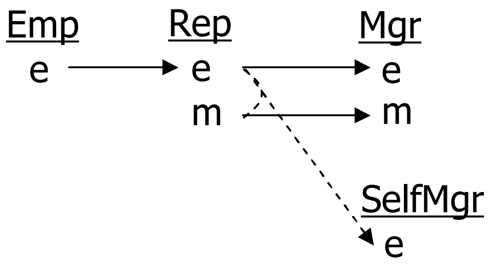
- If not, what is the "right" language for composing schema mappings?

Composition: Expressibility & Complexity

M_{12} Σ_{12}	M_{23} Σ_{23}	$M_{12} \circ M_{23}$ Σ_{13}	Composition Query
finite set of 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 $\varphi(\mathbf{x}) \rightarrow \exists\mathbf{y}\psi(\mathbf{x},\mathbf{y})$	in PTIME
finite set of s-t tgds $\varphi(\mathbf{x}) \rightarrow \exists\mathbf{y}\ \psi(\mathbf{x},\mathbf{y})$	finite set of (full) 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	in NP; can be NP-complete

Employee Example

- Σ_{12} :
  - Emp(e) $\rightarrow$ $\exists$m Rep(e,m)
- Σ_{23} :
  - Rep(e,m) $\rightarrow$ Mgr(e,m)
  - Rep(e,e) $\rightarrow$ 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:

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

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

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 schema mappings specified by SO-tgds, so that it produces universal solutions in polynomial time.
- For schema mappings specified by SO-tgds, the certain answers of target conjunctive queries are polynomial-time computable.

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 a "good" language for composing schema mappings.
  - SO-tgds are "chasable":
    Polynomial-time data exchange with universal solutions.
- SO-tgds and the composition algorithm have been incorporated in Criollo's Mapping Specification Language (MSL).

Related Work and Extensions in this PODS

- G. Gottlob:
  Computing Cores for Data Exchange: Algorithms & Practical Solutions

- A. Nash, Ph. Bernstein, S. Melnik:
  Composition of Mappings Given by Embedded Dependencies

- A. Fuxman, Ph. Kolaitis, R.J. Miller, W.-C. Tan:
  Peer Data Exchange

- M. Arenas & L. Libkin:
  XML Data Exchange: Consistency and Query Answering

Theory and Practice

"*Quelli che s'innamoran di pratica sanza scienza, son come 'l nocchiere ch'entra in navilio sanza timone o bussola, che mai ha certezza dove si vada*"

Leonardo da Vinci, 1452-1519

"*He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast*."

Reduction from 3-Colorability

- Σ_{12}
  - $\forall x \forall y\, (E(x,y) \rightarrow \exists u \exists v\, (C(x,u) \wedge C(y,v)))$
  - $\forall x \forall y\, (E(x,y) \rightarrow F(x,y))$
- Σ_{23}
  - $\forall x \forall y \forall u \forall v\, (C(x,u) \wedge C(y,v) \wedge F(x,y) \rightarrow D(u,v))$

- Let I_3 = { (r,g), (g,r), (b,r), (r,b), (g,b), (b,g) }

- Given **G**=(V, E),
  - let I_1 be the instance over $\mathbf{S}_1$ consisting of the edge relation E of **G**

- **G** is 3-colorable iff $\langle I_1, I_3 \rangle \in \mathrm{Inst}(M_{12}) \circ \mathrm{Inst}(M_{23})$

- [Dawar98] showed that 3-colorability is not expressible in $L_{\infty\omega}^{\omega}$

Algorithm Compose(M_{12}, M_{23})

- **Input**: Two schema mappings M_{12} and M_{23}
- **Output**: A schema mapping $M_{13} = M_{12} \circ M_{23}$

- Step 1: Split up tgds in Σ_{12} and Σ_{23}
  - C_{12} = Emp(e) $\rightarrow$ (Mgr1(e, f(e))
  - C_{23} =
    - Mgr1(e,m) $\rightarrow$ Mgr(e,m)
    - Mgr1(e,e) $\rightarrow$ SelfMgr(e)
- Step 2: Compose C_{12} with C_{23}
  - χ_1 : Emp(e_0) $\wedge$ ($e=e_0$) $\wedge$ ($m=f(e_0)$) $\rightarrow$ Mgr1(e,m)
  - χ_2 : Emp(e_0) $\wedge$ ($e=e_0$) $\wedge$ ($e=f(e_0)$) $\rightarrow$ SelfMgr(e)
- Step 3: Construct M_{13}
  - Return $M_{13} = (S_1, S_3, \Sigma_{13})$ where
  - $\Sigma_{13} = \exists f(\exists e_0 \exists e \exists m\ \chi_1 \wedge \exists e_0 \exists e\ \chi_2)$