Schema Mappings

&

Data Exchange

Phokion G. Kolaitis

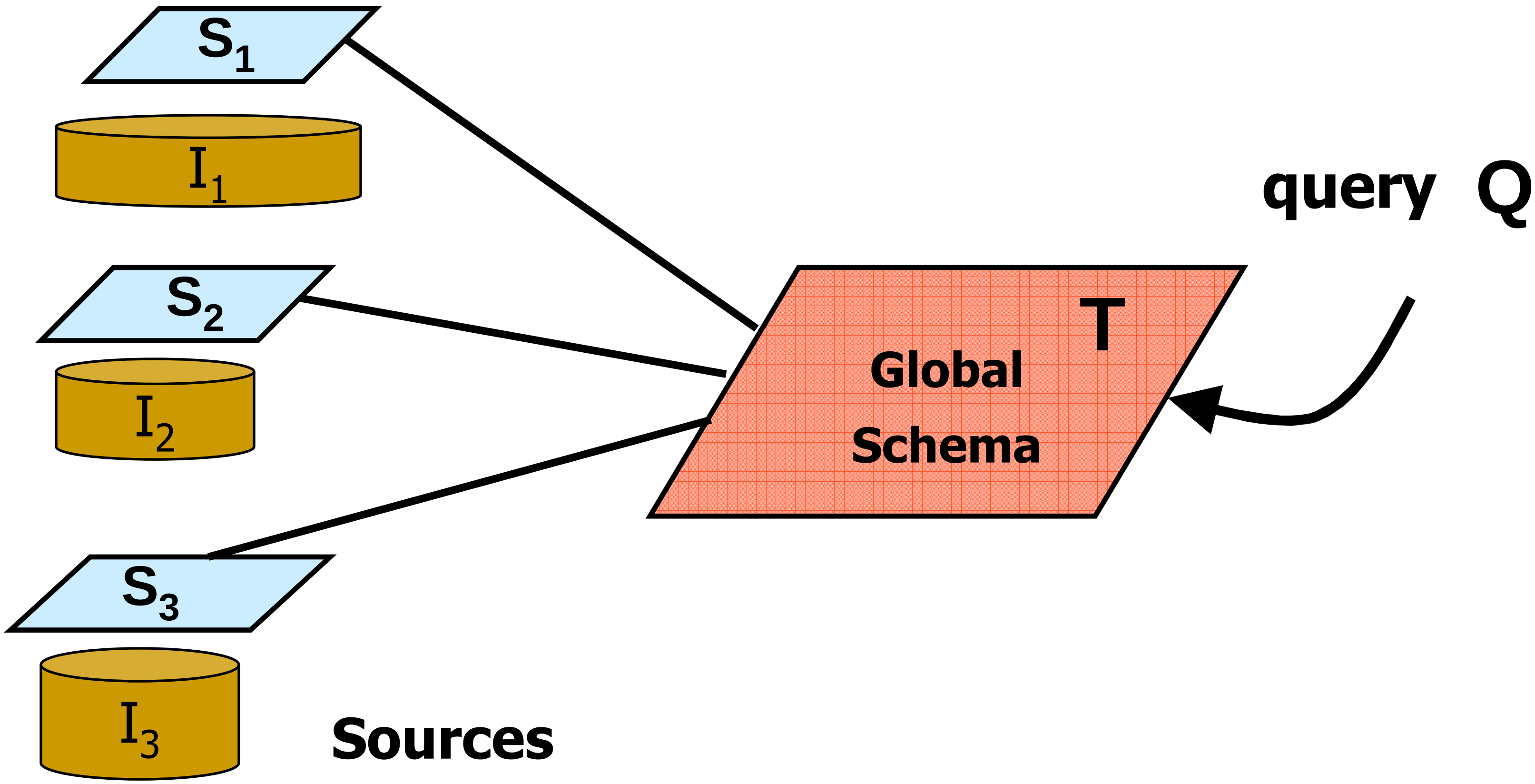
IBM Almaden Research Center

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**):

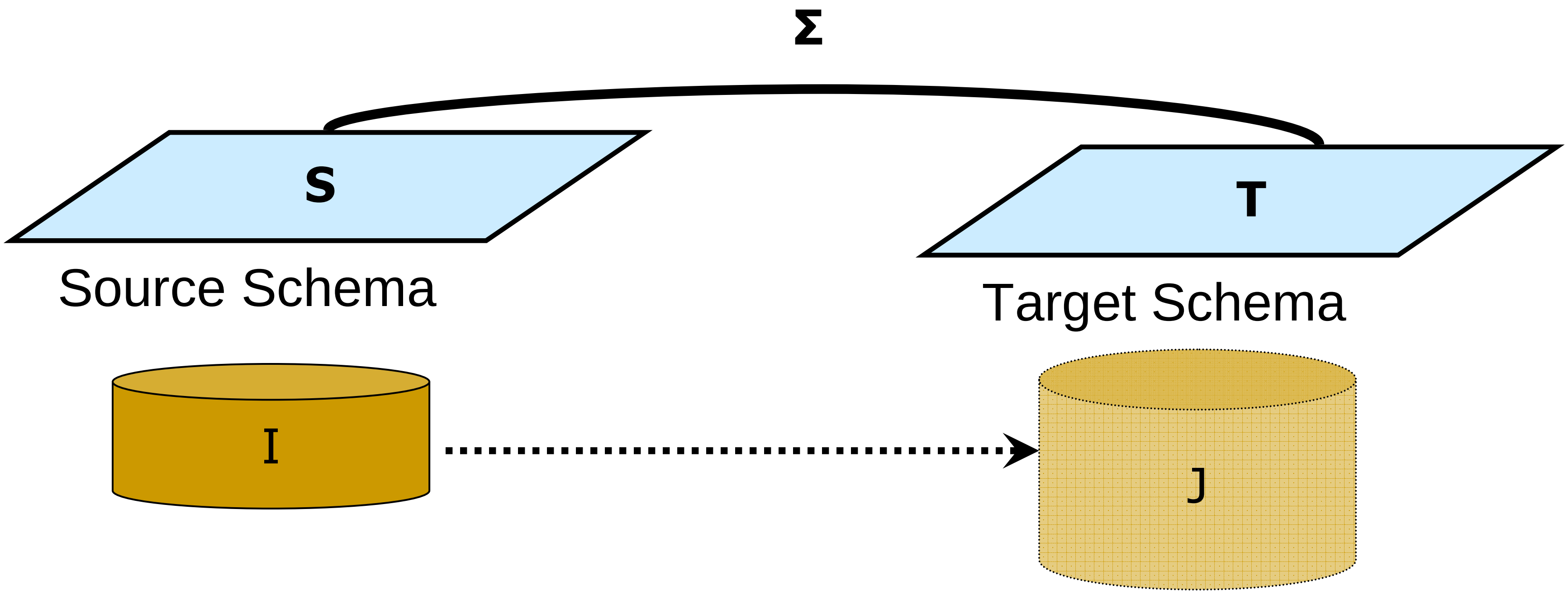
Data Integration

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 Course

- Schema Mappings and Data Exchange: Overview
- Conjunctive Queries and Homomorphisms
- Data Exchange with Schema Mappings Specified by Tgds and Egds
- Solutions in Data Exchange
  - Universal Solutions
  - Universal Solutions via the Chase
  - The Core of the Universal Solutions
- Query Answering in Data Exchange

Outline of the Course - continued

- Bernstein's Model Management Framework and Operations on Schema Mappings
- Composing Schema Mappings
- Inverting Schema Mapping
- Extensions of the Framework: Peer Data Exchange
- Open Problems and Research Directions

Credits

Much (but not all) of the material presented here is based on joint work with:

- Ron Fagin & Lucian Popa, IBM Almaden
- Ariel Fuxman (now at Microsoft Search Labs) & Renée J. Miller, U. of Toronto
- Jonathan Panttaja & Wang-Chiew Tan, UC Santa Cruz

and draws on papers in:

- ICDT ‘03, PODS ‘03, PODS ‘04, PODS ‘05, PODS ‘06
- TCS, ACM TODS

Basic Concepts: Relational Databases

- Relation Symbol: $R(A_1, \ldots, A_k)$

  R: relation name; $A_1, \ldots, A_k$ attribute names
- Schema:

  a sequence $\mathbf{S} = (R_1, \ldots, R_m)$ of relation symbols
- Instance (Relational Database) over **S**: a sequence

  $I = (R'_1, \ldots, R'_m)$ of relations (tables) such that

  arity (R_i) = arity (R'_i), for $i = 1, \ldots, m$.
- **Example:**
  - Relation Symbols:

    Enrolls(Student, Course), Teaches(Instructor, Course)
  - Schema: (Enrolls, Teaches)

Schema Mappings

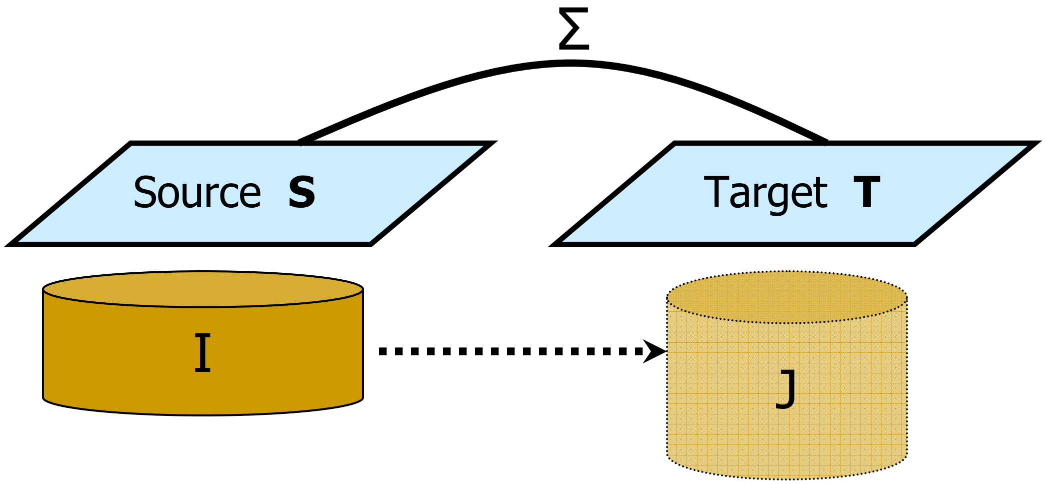
- Schema mappings:
  high-level, declarative assertions that specify the relationship between two schemas.

- 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 model 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.

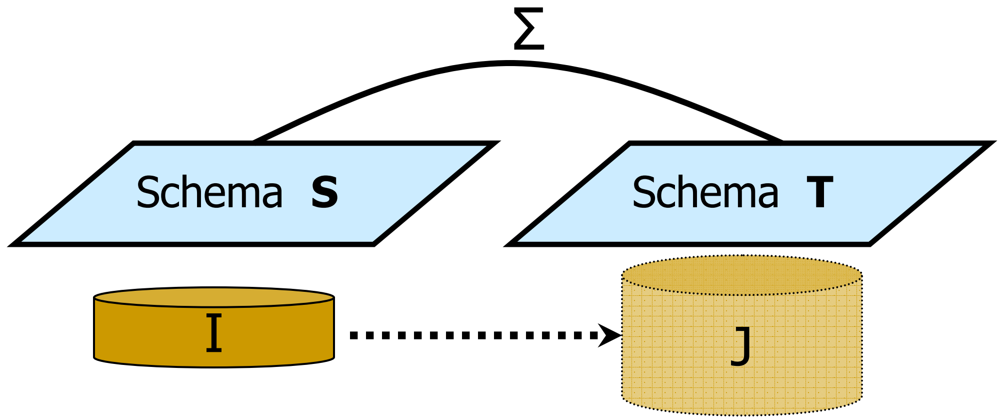
Fact: In general, for a given source instance I,

- No solution for I may exist (Σ overspecifies)

  or

- Multiple solutions for I may exist; in fact, infinitely many solutions for I may exist (Σ underspecifies).

Schema Mappings: Fundamental 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.

Queries

- **Definition:** Schema **S**
  - **k-ary query** Q on **S**-instances
    function $I \rightarrow Q(I)$ such that
    - $Q(I)$ is a k-ary relation on the active domain of I
    - Q is preserved under isomorphisms, i.e.,
      if $h: I \rightarrow J$ is an isomorphism, then $Q(J) = h(Q(I))$.
  - **Boolean query**: function $I \rightarrow Q(I) \in \{0,1\}$ and preserved under isomorphisms: $Q(J) = Q(I)$.
- **Example:**
  - Edge relation $E \rightarrow TC(E)$ (Transitive Closure; binary query)
  - Is E connected? (Boolean query)

Definability of Queries

- A k-ary query Q is **definable by a formula** $\phi(x_1, \ldots, x_k)$ if for all **S**-instances I

  $$Q(I) = \{(a_1, \ldots, a_k): I \models \phi(x_1/a_1, \ldots, x_k/a_k)\}$$

- A Boolean query Q is **definable by a sentence** ψ if for all **S**-instances I, we have that

  $$Q(I) = 1 \quad \text{if and only if } I \models \psi$$

Note: These are uniform definability notions
(the formula/sentence must work on all instances)

Conjunctive Queries

- **Definition:** A **conjunctive query** is a query definable by a FO-formula in prenex normal form built from atomic formula using $\exists$ and $\wedge$ only.

  $\exists z_1 \ldots \exists z_m \chi(x_{\mathbf{1}}, \ldots, x_k, z_1, \ldots, z_k)$

- **Examples:**
  - Path of Length 2: (binary query)
    - $\exists z (E(x,z) \wedge E(z,y))$
    - Written as a rule:
      - P(x,y) :-- E(x,z), E(z,y)
  - Cycle of Length 3: (Boolean query)
    - $\exists x \exists y \exists z (E(x,y) \wedge E(y,z) \wedge E(z,x))$
    - Written as a rule:
      - Q :-- E(x,z), E(z,y), E(z,x)

Conjunctive Queries

- Every **relational join** is a conjunctive query:

  P(A,B,C), R(B,C,D) two relation symbols

  $P \bowtie R$ (x,y,z,w) :-- P(x,y,z), R(y,z,w)

- Conjunctive queries are the most-frequently asked database queries; they are also known as **SPJ queries**

- The main construct of SQL expresses conjunctive queries:

  SELECT P.A, P.B, P.C, R.D

  FROM P, R

  WHERE P.B = R.B AND P.C = R.C

Conj. Query Evaluation and Containment

- **Definition:** Two fundamental problems about CQs
  - **Conjunctive Query Evaluation** (CQE):
    Given a conjunctive query Q and an instance I, find Q(I).

  - **Conjunctive Query Containment** (CQC):
    - Given two k-ary conjunctive queries Q_1 and Q_2, is it true that for every instance I, we have that $Q_1(I) \subseteq Q_2(I)$?
    - Given two Boolean queries Q_1 and Q_2, is it true that $Q_1 \models Q_2$? (that is, for all I, if $I \models Q_1$, then $I \models Q_2$)?
      CQC is logical implication.

CQE vs. CQC

Theorem: Chandra & Merlin, 1977

CQE and CQC are the ***same*** problem.

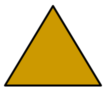
Question: What is the common link?

Answer: The **Homomorphism Problem**

Homomorphisms

- **Definition:** Let I and I′ be two instances over the same schema.
  A **homomorphism** h: I $\rightarrow$ I′ is a function from the active domain of I to the active domain of I′ such that
  if $P(a_1,\ldots,a_m)$ is in I, then $P(h(a_1),\ldots,h(a_m))$ is in I′.

- **Definition: The Homomorphism Problem**
  Given two instances I and I′, is there a homomorphism h: I $\rightarrow$ I′?

- **Examples:**
  - A graph G = (V,E) is 3-colorable
    if and only if
    there is a homomorphism h: G $\rightarrow$ K_3 
  - 3-SAT can be viewed as a Homomorphism Problem

Canonical CQs and Canonical Instances

- **Definition:** Canonical Conjunctive Query

  Given an instance $I = (R_1, \ldots, R_m)$, the **canonical CQ** of I is the Boolean conjunctive query Q^I with the elements of I as variables and the facts of I as conjuncts.

- **Example:**

  I consists of E(a,b), E(b,c), E(c,a)

  - Q^I is given by the rule:

    Q^I :-- E(x,z), E(z,y), E(z,x)

  - Alternatively, Q^I is

    $$\exists x \, \exists y \, \exists z \, (E(x,z) \wedge E(z,y) \wedge E(z,x))$$

Canonical Databases

- **Definition:** Canonical Instance

  Given a Boolean CQ Q, the **canonical instance** of Q is the instance I^Q with the variables of Q as elements and the conjuncts of Q as facts.

- **Example:**

  Conjunctive query Q :-- E(x,y),E(x,z)

  Canonical instance I^Q consists of the facts E(x,y), E(x,z)

Homomorphisms, CQE, and CQC

Theorem: Chandra & Merlin – 1977

For instances I and I', the following are equivalent:

- There is a homomorphism h: $I \to I'$
- $I' \models Q^I$
- $Q^{I'} \subseteq Q^I$

In dual form:

Theorem: Chandra & Merlin – 1977

For CQs Q and Q′, the following are equivalent:

- $Q \subseteq Q'$
- There is a homomorphism h: $I^{Q'} \to I^Q$
- $I^Q \models Q'$.

Illustrating the Chandra-Merlin Theorem

Example: 3-Colorability

For a graph G=(V,E), the following are equivalent:

- G is 3-colorable
- There is a homomorphism h: G $\rightarrow$ K_3
- $K_3 \models Q^G$
- $Q^{K^3} \subseteq Q^G$.

Combined complexity of CQC and CQE

Corollary: The following problems are NP-complete:

- Given two conjunctive queries Q and Q′ is $Q \subseteq Q'$?
- Given a conjunctive query Q and an instance I, does $I \models Q$?

Proof:

(a) Membership in NP follows from Chandra & Merlin:

$Q \subseteq Q'$ iff there is a homomorphism $h: I^{Q'} \rightarrow I^{Q}$

(b) NP-hardness follows from 3-Colorability.

Combined Complexity vs. Data Complexity

Vardi's Taxonomy of Query Evaluation (1982):

- **Combined Complexity:** Both the query and the instance are part of the input.
- **Data Complexity:** Fix the query; the input consists of the instance only.

Complexity of Conjunctive Queries:

- The combined complexity of conjunctive queries is NP-complete.
- For each fixed conjunctive query Q, the data complexity of Q is in P (in fact, it is in LOGSPACE).

Course Outline – Progress Report

- ✓ Schema Mappings and Data Exchange: Overview
- ✓ Conjunctive Queries and Homomorphisms
- Data Exchange with Schema Mappings Specified by Tgds and Egds
- Solutions in Data Exchange
  - Universal Solutions
  - Universal Solutions via the Chase
  - The Core of the Universal Solutions
- Query Answering in Data Exchange

Embedded Implicational Dependencies

- Dependency Theory: extensive study of constraints in relational databases in the 1970s and 1980s.
- Conjunctive queries are used as building blocks in specifying constraints in relational databases.
- 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
  in ICDT 2003 and TCS

- 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)

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

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

Fact: Every s-t tgd asserts that the result of a CQ over the source is contained in the result of a CQ over the target.

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

Schema Mapping Specification Language

- From now on, we will drop the universal quantifiers in the front. So, instead of $\forall \mathbf{x} \forall \mathbf{x'} (\varphi(\mathbf{x}, \mathbf{x'}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}))$,
  we will write $(\varphi(\mathbf{x}, \mathbf{x'}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}))$.

- **Example:**

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

  This s-t tgd asserts that the result of the conjunctive query

  $\exists$ y (Student(s) $\wedge$ Enrolls(s,c,y))

  is contained in the resut of the conjunctive query

  $\exists$t $\exists$g (Teaches(t,c) $\wedge$ Grade(s,c,g)).

Schema Mapping Specification Language

- **Full tgds** are tgds of the form

$$\phi(\mathbf{x},\mathbf{x'}) \rightarrow \psi(\mathbf{x}),$$

where $\phi(\mathbf{x})$ and $\psi(\mathbf{x})$ are conjunctions of atoms (no existential quantifiers in the right-hand side)

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

- Full tgds of the form

$$\phi(\mathbf{x}) \rightarrow \psi(\mathbf{x})$$

express the containment between two relational joins.

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

- **Note:** Full tgds have "good" algorithmic properties in data exchange.

Constraints in Data Integration

Fact: 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.

Note:

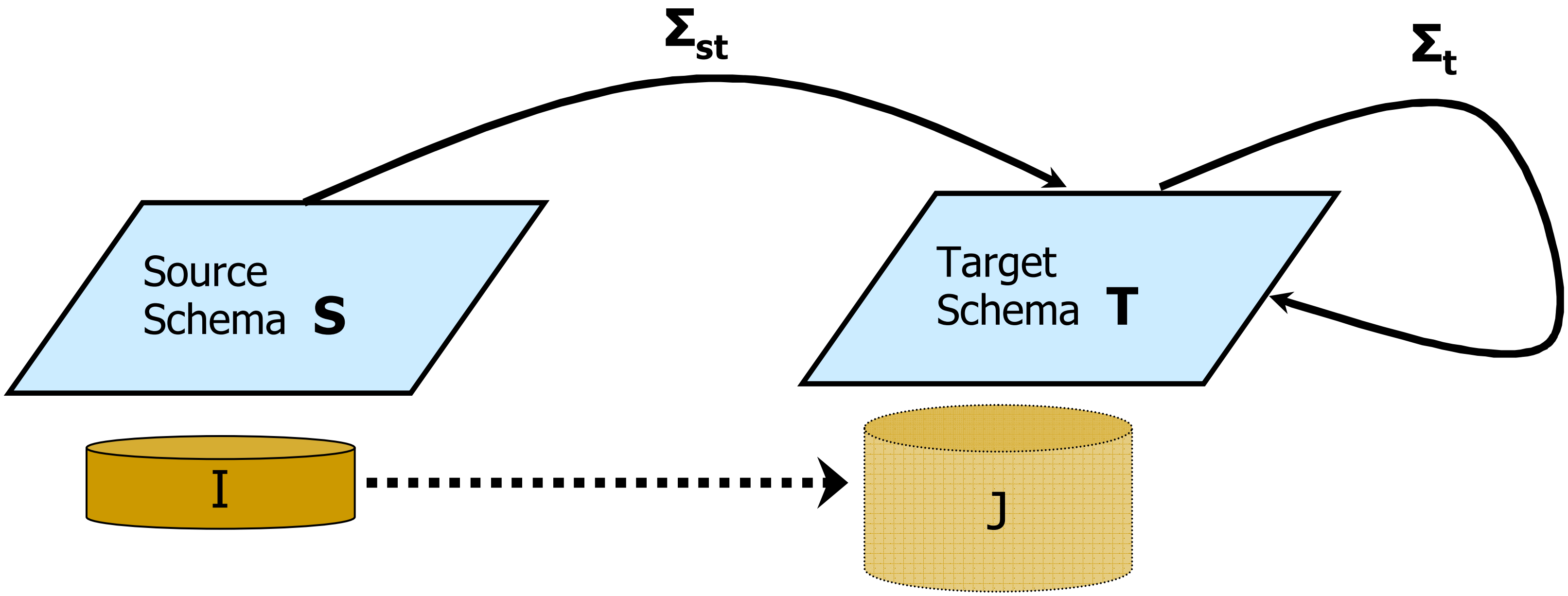
At present, most commercial II systems support GAV only.

Target Dependencies

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

- Target Tgds : $\varphi_T(\mathbf{x},\mathbf{x'}) \rightarrow \exists \mathbf{y}\ \psi_T(\mathbf{x}, \mathbf{y})$
  - Dept (did, dname, mgr_id, mgr_name) $\rightarrow$ Mgr (mgr_id, did)
    (a target inclusion dependency constraint)
  - $F(x,y) \wedge F(y,z) \rightarrow F(x,z)$
- Target Equality Generating Dependencies (egds):
  $$\varphi_T(\mathbf{x}) \rightarrow (x_1=x_2)$$
  - (Mgr (e, d_1) $\wedge$ 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

Algorithmic Problems in Data Exchange

Definition: Schema Mapping **M** = (**S**, **T**, Σ_{st},Σ_t),
If I is a source instance, then a solution for I is a target instance J such that <I, J > satisfy $\Sigma_{st} \cup \Sigma_t$.

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

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?

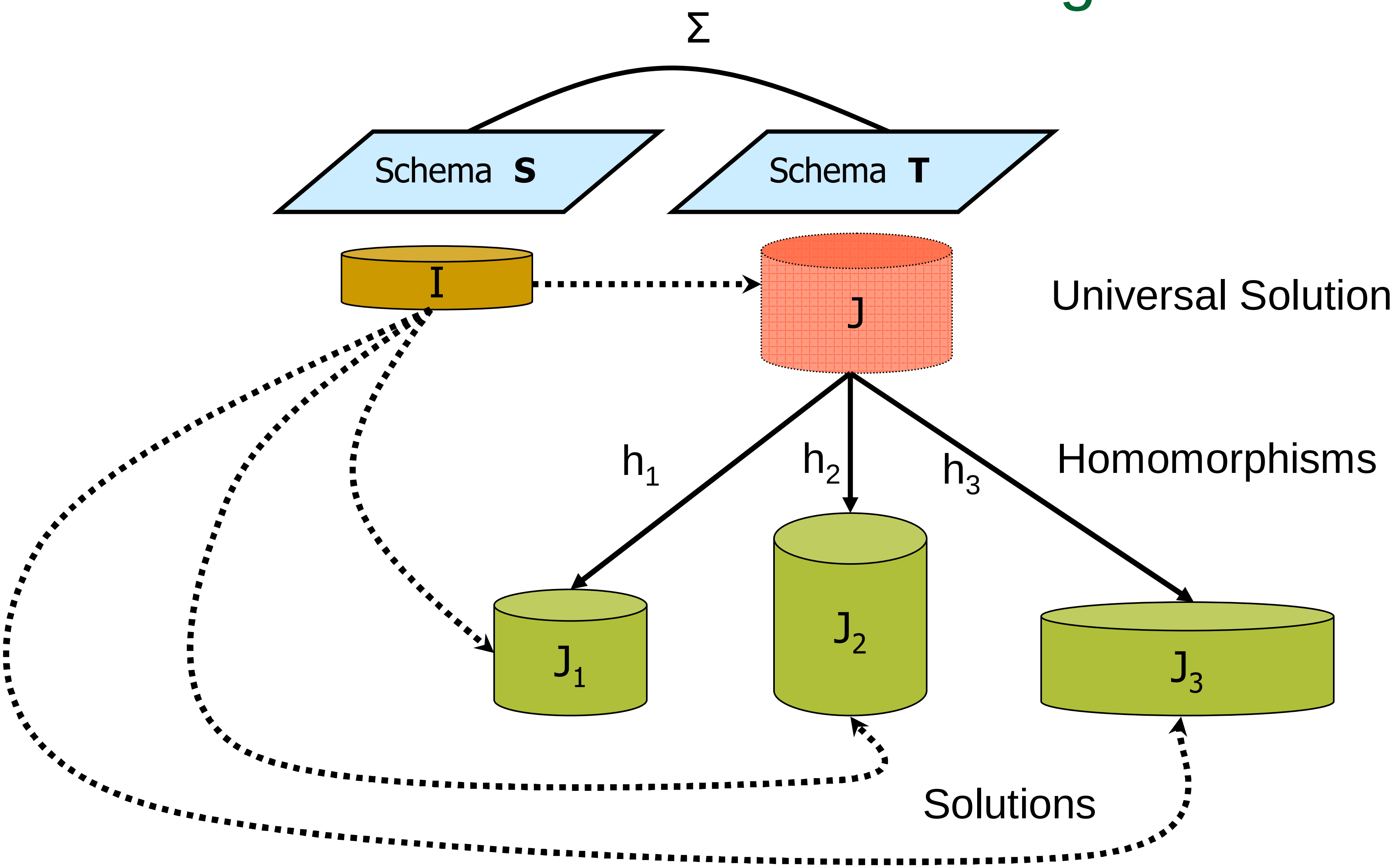
Universal Solutions in Data Exchange

We introduced the notion of universal solutions as the "best"solutions in data exchange.

Definition: a solution is **universal** if it has homomorphisms that preserve constants 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 $I = \{E(a,b)\}$

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.

The Existence-of-Solutions Problem

Question: What can we say about the existence-of-solutions problem **Sol(M)** for a fixed schema mapping **M** = (**S**, **T**, Σ_{st},Σ_t) specified by s-t tgds and target tgs and egds?

Fact: Depending on the target constraints in Σ_t,

- **Sol(M)** can be trivial (solutions always exist).
- …
- **Sol(M)** can be in PTIME.
- …
- **Sol(M)** can be undecidable.

Algorithmic Problems in Data Exchange

Proposition: If **M** = (**S**, **T**, Σ_{st},Σ_t) is a schema mapping such that Σ_t is a set of **full target tgds**, then:

- Solutions always exist; hence, **Sol(M)** is trivial.
- There is a **Datalog program** π over the target **T** that can be used to compute universal solutions as follows:
  Given a source instance I,
  1. Compute a universal solution J* for I w.r.t. the schema mapping **M*** = (**S**, **T**, Σ_{st}) using the **naïve chase** algorithm.
  2. Run the **Datalog program** π on J* to obtain a universal solution J for I w.r.t. **M**.
- Consequently, universal solutions can be computed in polynomial time.

Algorithmic Problems in Data Exchange

Naïve chase algorithm 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.

Example: **M** = (**S**, **T**, Σ_{st},Σ_t)

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

Σ_t: $F(u,w) \wedge F(w,v) \rightarrow F(u,v)$

1. The naïve chase returns a relation F* obtained from E by adding a new node between every edge of E.
2. The Datalog program π computes the **transitive closure** of F*.

Algorithmic Problems in Data Exchange

Fact: If **M** = (**S**, **T**, Σ_{st},Σ_t) is a schema mapping such that Σ_t is a set of **full target tgds**, then

- Solutions always exist; hence, **Sol(M)** is trivial.
- There is a **Datalog program** π over the target **T** that can be used to compute universal solutions as follows:

  Given a source instance I,

  1. Compute a universal solution J for I w.r.t. the schema mapping **M** = (**S**, **T**, Σ_{st}) using the **naïve chase**.

  2. Run the **Datalog program** π on J.

Consequently, universal solutions can be computed in polynomial time.

Algorithmic Problems in Data Exchang

Fact: If **M** = (**S**, **T**, Σ_{st},Σ_t) is a schema mapping such that Σ_t is a set of **full target tgds** and **target egds, then:**

- Solutions need not always exist.
- The existence-of-solutions problem **Sol(M)** may be P-complete.

Proof: Reduction from Horn 3-SAT.

Algorithmic Problems in Data Exchange

Reducing Horn 3-SAT to the Existence-of-Solutions Problem **Sol(M)**

- Σ_{st}:
  U(x) $\rightarrow$ U′(x)
  P(x,y,z) $\rightarrow$ P′(x,y,z)
  N(x,y,z) $\rightarrow$ N′(x,y,z)
  V(x) $\rightarrow$ V′(x)

- Σ_t:
  U′(x) $\rightarrow$ M′(x)
  P′(x,y,z) $\wedge$ M′(y) $\wedge$ M′(z) $\rightarrow$ M′(x)
  N′(x,y,z) $\wedge$ M′(x) $\wedge$ M′(y) $\wedge$ M′(z) $\wedge$ V′(u) $\rightarrow$ W′(u)
  W′(u) $\wedge$ W′(v) $\rightarrow$ u = v

- U(x) encodes the unit clause x
  P(x,y,z) encodes the clause ($\neg$ y $\vee$ $\neg$ z $\vee$ x)
  N(x,y,z) encodes the clause ($\neg$ x $\vee$ $\neg$ y $\vee$ $\neg$ z)
  V = {0, 1}

Algorithmic Problems in Data Exchange

Question:

What about arbitrary target tgds and egds?

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 source-to-target tgd;
- Σ^*_t consists of one egd, one full target tgd, and one (non-full) target tgd;
- 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?

The Embedding Problem & Data Exchange

- **Theorem (Evans – 1950s):**

  K class of algebras closed under isomorphisms.

  The following are equivalent:
  - The word problem for ***K*** is decidable.
  - The embedding problem for ***K*** is decidable.

- **Theorem (Gurevich – 1966):**

  The word problem for finite semigroups is undecidable.

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
    $R'(x,y,z) \wedge R'(x',y',z') \rightarrow \exists w_1 \ldots \exists w_9$
    $(R'(x,x',w_1) \wedge R'(x,y',w_2) \wedge R'(x,z',w_3)$
    $R'(y,x',w_4) \wedge R'(y,y',w_5) \wedge R'(x,z',w_6)$
    $R'(z,x',w_7) \wedge R'(z,y',w_8) \wedge R'(z,z',w_9))$

The Existence-of-Solutions Problem

Summary: The existence-of-solutions problem

- is **undecidable** for schema mappings in which the target dependencies are arbitrary tgds and egds;
- is in P for schema mappings in which the target dependencies are **full** tgds and egs.

Question: Are classes of target tgds **richer** than full tgds and and egds for which the existence-of-solutions problem is in P?

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 Set of Tgds

- The concept of **weakly acyclic set of tgds** was formulated by Alin Deutsch and Lucian Popa.
- It was first used independently by Deutsch and Tannen and by FKMP in papers that appeared in ICDT 2003.
- Weak acyclicity is a fairly broad structural condition: it contains as special cases several other concepts studied earlier.

Weakly Acyclic Sets of Tgds

Weakly acyclic sets of tgds contain as special cases:

- **Sets of full tgds**

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

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

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

- **Acyclic sets of inclusion dependencies**

Large class of dependencies occurring in practice.

Weakly Acyclic Sets of Tgds: Definition

- **Dependency 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 as (R,A):
    - For every occurrence of x in ψ as (S,B), add an edge (R,A) $\longrightarrow$ (S,B)
    - In addition, for every existentially quantified y that occurs in ψ as (T,C), add a **special edge** (R,A) $\Longrightarrow$ (T,C).

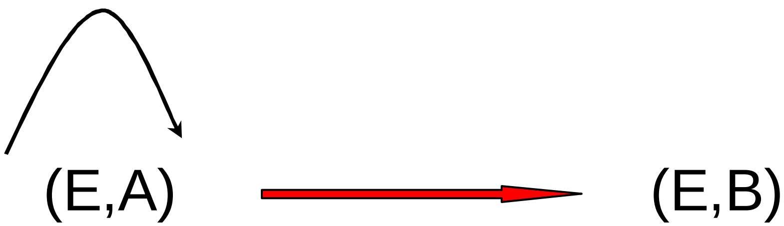
- Σ is **weakly acyclic** if the dependency graph has **no** cycle containing a **special edge**.

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

Weakly Acyclic Sets of Tgds: Examples

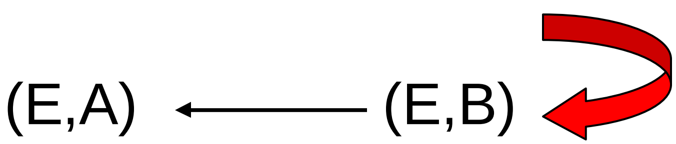
- **Example 1:**

  $E(x,y) \rightarrow \exists z\ E(x,z)$ is weakly acyclic



- **Example 2:**

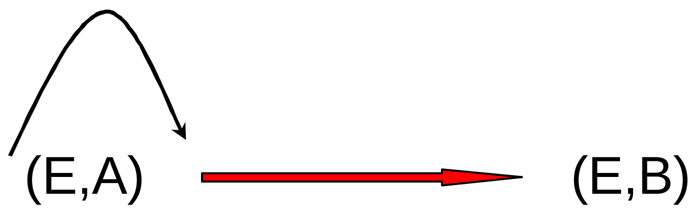
  $E(x,y) \rightarrow \exists z\ E(y,z)$ is not weakly acyclic



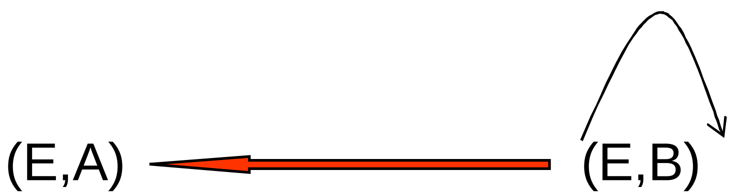
Weakly Acyclic Sets of Tgds: Examples

Example 3: Weak Acyclicity is **not** preserved under unions

- $E(x,y) \rightarrow \exists z\, E(x,z)$ is weakly acyclic



- $E(x,y) \rightarrow \exists z\, E(z,y)$ is weakly acyclic



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

Weakly Acyclic Sets of Tgds: Examples

- **Example 3:** The target tgd

$$R'(x,y,z) \wedge R'(x',y',z') \rightarrow \exists w_1 \ldots \exists w_9$$
$$(R'(x,x',w_1) \wedge R'(x,y',w_2) \wedge R'(x,z',w_3)$$
$$R'(y,x',w_4) \wedge R'(y,y',w_5) \wedge R'(x,z',w_6)$$
$$R'(z,x',w_7) \wedge R'(z,y',w_8) \wedge R'(z,z',w_9))$$

is not weakly acyclic **(Why?)**

Data Exchange with Weakly Acyclic Tgds

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.

There is an algorithm, based on the chase procedure, so that:

- Given a source instance I, the algorithm determines if a solution for I exists; if so, it produces a canonical universal solution for I.
- The running time of the algorithm is polynomial in the size of I.
- Hence**,** the existence-of-solutions problem **Sol(M)** for **M**, is in P.

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".

Weak Acyclicity and the Chase Procedure

Note: If the set of target tgds is not weakly acyclic, then the chase 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

The Complexity of Data Exchange

- The results presented thus far assume that the schema mapping is kept **fixed**, while the source instance **varies**.
- In Vardi's taxonomy, this means all preceding results are about the **data complexity** of data exchange.

- **Question:**
  - Do the results change if both the schema mapping and the source instance are part of the input to the existence-of-solutions problem? If so, how do they change?
  - In other words, what is the **combined complexity** of data exchange?

The Existence-of-Solutions Problem

Proposition: Let **M** = (**S**, **T**, Σ_{st},Σ_t) be a schema mapping such that $\Sigma_t = \emptyset$ (no target constraints). Then

- **Sol(M)** is trivial (for every source instance, there is a solution).
- Universal solutions can be constructed in polynomial time.

Proof: Use a **naïve chase** algorithm: 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.

The Existence-of-Solutions Problem

Example 1: Collapsing paths of length 2 to edges

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

- I_1 = { E(1,3}, E(2,4), E(3,4) }
  J_1 = { F(1,4) } universal solution for I_1

- I_2 = { E(1,3}, E(2,4), E(3,4), E(4,3)}
  J_2 = { F(1,4), F(2,3), F(3,3) } universal solution for I_2

The Existence-of-Solutions Problem

Example 2: Transforming edges to paths of length 2

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

- I_1 = { E(1,2) }
  J_1 = { F(1,X), F(X,2) } universal solution for I_1
- I_2 = { E(1,2}, E(3,4)}
  J_2 = { F(1,X), F(X,2), F(3,Y), F(Y,4) } universal solution for I_2

Algorithmic Problems in Data Exchange

Fact: If **M** = (**S**, **T**, Σ_{st},Σ_t) is a schema mapping such that Σ_t is a set of **full target tgds**, then

- Solutions always exist; hence, **Sol(M)** is trivial.
- There is a **Datalog program** π over the target **T** that can be used to compute universal solutions as follows:

  Given a source instance I,

  1. Compute a universal solution J for I w.r.t. the schema mapping **M** = (**S**, **T**, Σ_{st}) using the **naïve chase**.

  2. Run the **Datalog program** π on J.

Consequently, universal solutions can be computed in polynomial time.

Algorithmic Problems in Data Exchange

Example:

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

Σ_t: $F(u,w) \wedge F(w,v) \rightarrow F(u,v)$

1. The naïve chase returns a relation F* obtained from E by adding a new node between every edge of E.
2. The Datalog program computes the **transitive closure** of F*.

Datalog

“ Datalog = Conjunctive Queries + Recursion ”

Definition: A **Datalog program** π is a finite set of rules each expressing a conjunctive query.

Example: **Transitive Closure**

P(x,y) :-- E(x,y)
P(x,y) :-- E(x,z), P(z,y)

Note: A relation symbol may occur both in the head and in the body of a rule.

Datalog

Example 1: Paths of Odd and Even Length

```
ODD(x,y)   :--   E(x,y)
ODD(x,y)   :--   E(x,z), EVEN(z,y)
EVEN(x,y) :--   E(x,z), ODD(z,y).
```

Example 2: Non 2-Colorability

```
ODD(x,y)   :--   E(x,y)
ODD(x,y)   :--   E(x,z), EVEN(z,y)
EVEN(x,y) :--   E(x,z), ODD(z,y).
 Q              :--   ODD(x,x)
```

Datalog Semantics

- **Procedural Semantics:**

  Bottom-up evaluation of recursive predicates (IDBs)

  1. Set all recursive to $\emptyset$.
  2. Apply all rules in parallel; update the recursive predicates.
  3. Repeat until no recursive predicate changes.

- **Declarative Semantics:**

  Least fixed-point of an existential positive FO-formula extracted from the program.

$$\phi(x,y,P):\ E(x,y) \vee \exists z\,(E(x,z) \wedge P(z,y))$$

Complexity of Datalog

Fact:

- **Data Complexity of Datalog:**
  Every fixed Datalog program can be evaluated in polynomial-time.

  Reason: Bottom-up evaluation converges in polynomially-many steps.

- **Combined Complexity of Datalog:**
  EXPTIME-complete.

Complexity of Datalog

Fact: The data complexity of Datalog can be P-complete.

Proof: **Path Systems Problem**

T(x) :-- A(x)
T(x) :-- R(x,y,z), T(y), T(z)

Cook (1974) has shown that evaluating this Datalog program is P-complete.

Algorithmic Problems in Data Exchange

Fact: If **M** = (**S**, **T**, Σ_{st},Σ_t) is a schema mapping such that Σ_t is a set of **full target tgds**, then

- Solutions always exist; hence, **Sol(M)** is trivial.
- There is a **Datalog program** π over the target **T** that can be used to compute universal solutions as follows:

  Given a source instance I,

  1. Compute a universal solution J for I w.r.t. the schema mapping **M** = (**S**, **T**, Σ_{st}) using the **naïve chase**.

  2. Run the **Datalog program** π on J.

Consequently, universal solutions can be computed in polynomial time.

Algorithmic Problems in Data Exchang

Fact: If **M** = (**S**, **T**, Σ_{st},Σ_t) is a schema mapping such that Σ_t is a set of **full target tgds** and **target egds, then:**

- Solutions need not always exist.
- The existence-of-solutions problem **Sol(M)** may be P-complete.

Proof: Reduction from Horn 3-SAT.

Algorithmic Problems in Data Exchange

Reducing Horn 3-SAT to the Existence-of-Solutions Problem **Sol(M)**

- Σ_{st}:
  $U(x) \rightarrow U'(x)$
  $P(x,y,z) \rightarrow P'(x,y,z)$
  $N(x,y,z) \rightarrow N'(x,y,z)$
  $V(x) \rightarrow V'(x)$

- Σ_t:
  $U'(x) \rightarrow M'(x)$
  $P'(x,y,z) \wedge M'(y) \wedge M'(z) \rightarrow M'(x)$
  $N'(x,y,z) \wedge M'(x) \wedge M'(y) \wedge M'(z) \wedge V'(u) \rightarrow W'(u)$
  $W'(u) \wedge W'(v) \rightarrow u = v$

- $U(x)$ encodes the unit clause x
  $P(x,y,z)$ encodes the clause $(\neg y \vee \neg z \vee x)$
  $N(x,y,z)$ encodes the clause $(\neg x \vee \neg y \vee \neg z)$
  $V = \{0, 1\}$

Algorithmic Problems in Data Exchange

Question:

What about arbitrary target tgds and egds?

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 source-to-target tgd;
- Σ^*_t consists of one egd, one full target tgd, and one (non-full) target tgd;
- 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?

The Embedding Problem & Data Exchange

- **Theorem (Evans – 1950s):**

  K class of algebras closed under isomorphisms.

  The following are equivalent:

  - The word problem for ***K*** is decidable.
  - The embedding problem for ***K*** is decidable.

- **Theorem (Gurevich – 1966):**

  The word problem for finite semigroups is undecidable.

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
    $$R'(x,y,z) \wedge R'(x',y',z') \rightarrow \exists w_1 \ldots \exists w_9$$
    $$(R'(x,x',w_1) \wedge R'(x,y',w_2) \wedge R'(x,z',w_3)$$
    $$R'(y,x',w_4) \wedge R'(y,y',w_5) \wedge R'(x,z',w_6)$$
    $$R'(z,x',w_7) \wedge R'(z,y',w_8) \wedge R'(z,z',w_9))$$

The Existence-of-Solutions Problem

Summary: The existence-of-solutions problem

- is **undecidable** for schema mappings in which the target dependencies are arbitrary tgds and egds;
- is in P for schema mappings in which the target dependencies are **full** tgds and egs.

Question: Are classes of target tgds **richer** than full tgds and and egds for which the existence-of-solutions problem is in P?

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 Set of Tgds

- The concept of **weakly acyclic set of tgds** was formulated by Alin Deutsch and Lucian Popa.

- It was first used independently by Deutsch and Tannen and by FKMP in papers that appeared in ICDT 2003.

- Weak acyclicity is a fairly broad structural condition: it contains as special cases several other concepts studied earlier.

Weakly Acyclic Sets of Tgds

Weakly acyclic sets of tgds contain as special cases:

- **Sets of full tgds**

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

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

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

- **Acyclic sets of inclusion dependencies**

Large class of dependencies occurring in practice.

Weakly Acyclic Sets of Tgds: Definition

- **Dependency 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 as (R,A):
    - For every occurrence of x in ψ as (S,B), add an edge (R,A) $\longrightarrow$ (S,B)
    - In addition, for every existentially quantified y that occurs in ψ as (T,C), add a **special edge** (R,A) $\Longrightarrow$ (T,C).

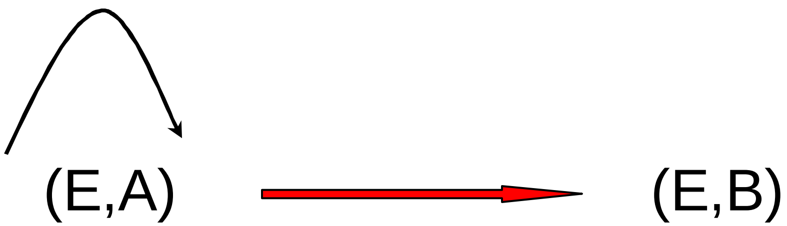
- Σ is **weakly acyclic** if the dependency graph has **no** cycle containing a **special edge**.

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

Weakly Acyclic Sets of Tgds: Examples

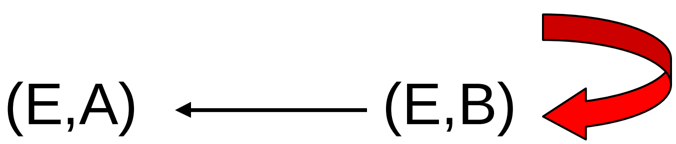
- **Example 1:**

  $E(x,y) \rightarrow \exists z\, E(x,z)$ is weakly acyclic



- **Example 2:**

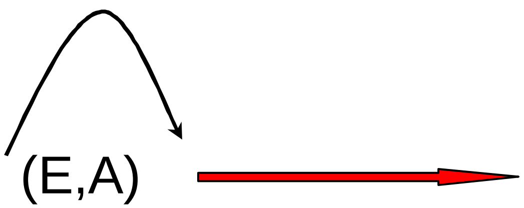
  $E(x,y) \rightarrow \exists z\, E(y,z)$ is not weakly acyclic



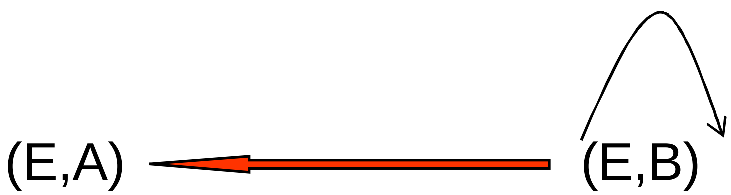
Weakly Acyclic Sets of Tgds: Examples

Example 3: Weak Acyclicity is **not** preserved under unions

- $E(x,y) \rightarrow \exists z\, E(x,z)$ is weakly acyclic



- $E(x,y) \rightarrow \exists z\, E(z,y)$ is weakly acyclic



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

Weakly Acyclic Sets of Tgds: Examples

- **Example 3:** The target tgd

$$R'(x,y,z) \wedge R'(x',y',z') \rightarrow \exists w_1 \ldots \exists w_9$$
$$(R'(x,x',w_1) \wedge R'(x,y',w_2) \wedge R'(x,z',w_3)$$
$$R'(y,x',w_4) \wedge R'(y,y',w_5) \wedge R'(x,z',w_6)$$
$$R'(z,x',w_7) \wedge R'(z,y',w_8) \wedge R'(z,z',w_9))$$

is not weakly acyclic **(Why?)**

Data Exchange with Weakly Acyclic Tgds

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.

There is an algorithm, based on the chase procedure, so that:

- Given a source instance I, the algorithm determines if a solution for I exists; if so, it produces a canonical universal solution for I.
- The running time of the algorithm is polynomial in the size of I.
- Hence**,** the existence-of-solutions problem **Sol(M)** for **M**, is in P.

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".

Weak Acyclicity and the Chase Procedure

Note: If the set of target tgds is not weakly acyclic, then the chase 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

The Complexity of Data Exchange

- The results presented thus far assume that the schema mapping is kept **fixed**, while the source instance **varies**.
- In Vardi's taxonomy, this means all preceding results are about the **data complexity** of data exchange.

- **Question:**
  - Do the results change if both the schema mapping and the source instance are part of the input to the existence-of-solutions problem? If so, how do they change?
  - In other words, what is the **combined complexity** of data exchange?

Combined Complexity of Data Exchange

Theorem (K …, Panttaja, Tan): **M** = (**S**, **T**, Σ_{st}, Σ_t) such that Σ_t is the union of a **weakly acyclic set** of target tgds with a set of target egds.

- The combined complexity of **Sol(M)** is 2EXPTIME-complete.
- **If S** and **T** are kept fixed**,** the combined complexity of **Sol(M)** is EXPTIME-complete.
- If **S** and **T** are kept fixed and Σ_t is the union of a set of **full** target tgds with a set of target egds, the combined complexity of **Sol(M)** is coNP-complete.

Hint of Proof:

- 2EXPTIME-hardness is via a reduction from EXPSPACE ATMs.
- EXPTIME-hardness is via a reduction from the combined complexity of **Datalog single-rule programs**

  Gottlob & Papadimitriou – 2003.

The Complexity of Data Exchange

	Schema Mapping **M**	**Sol(M)**
Data Complexity	Fixed; arbitrary target tgds	Can be undecidable
	Fixed; weakly acyclic target tgds and egds	In P; can be P-complete
Combined Complexity	Varies; weakly acyclic target tgds & egds	2EXPTIME-complete
	Fixed Schemas; Σ_{st}, and Σ_t vary; weakly acyclic target tgds & egds	EXPTIME-complete
	Fixed Schemas; Σ_{st}, and Σ_t vary; full target tgds & egds	coNP-complete

The Smallest Universal Solution

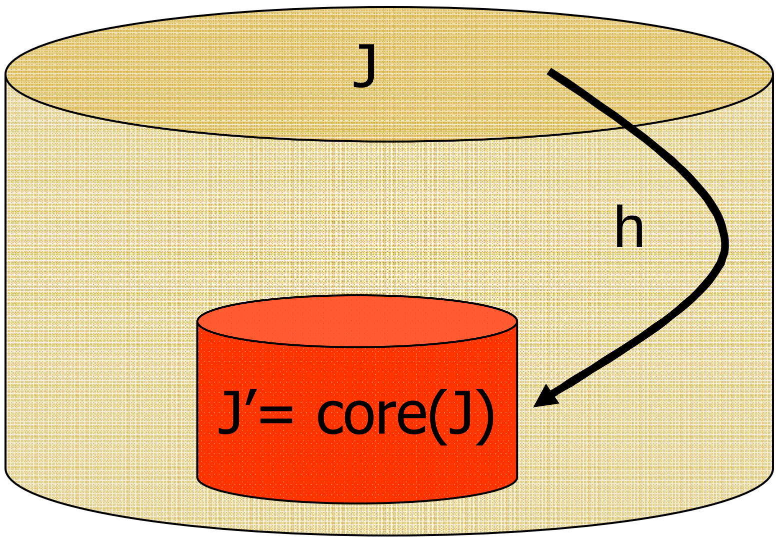
- **Fact:** Universal solutions need not be unique.
- **Question**: Is there a "best" universal solution?
- **Answer:** In joint work with R. Fagin and L. Popa, we took a "small is beautiful" approach:

  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.

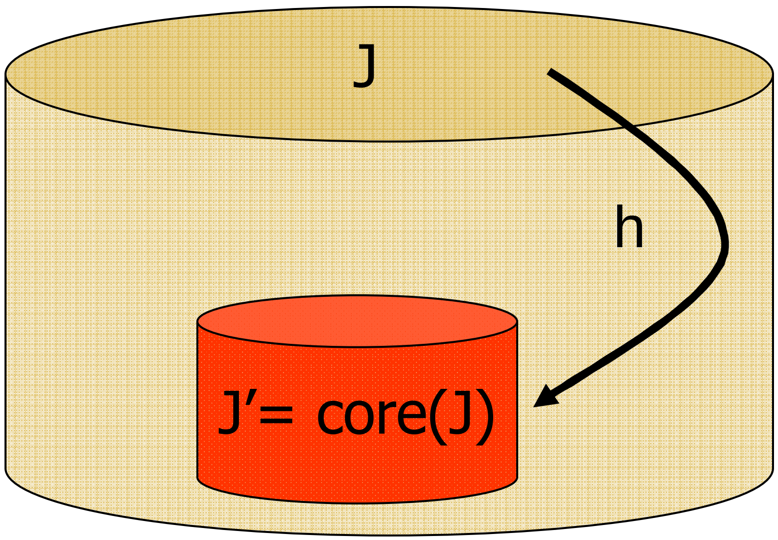
The Core of a Structure



Definition: J′ is the core of J if

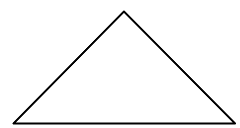
- J′ ⊆ J
- there is a hom. h: J → J′
- there is **no** hom. g: J → J″, where J″ ⊂ J′.

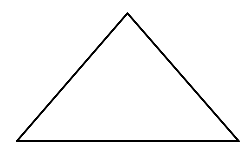
The Core of a Structure



Definition: J' is the core of J if

- $J' \subseteq J$
- there is a hom. $h: J \rightarrow J'$
- 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.

Complexity of the Core in Graph Theory

Theorem: Hell & Nesetril – 1992

Core Recognition is coNP-complete: given graph G, is G a core?

Theorem: (FKP)

Core Identification is DP-complete:

given graphs G and H, is H the core of G?

Definition: Papadimitriou & Yannakakis – 1982

DP is the class of all decision problem that can be written as
the conjunction of an NP-problem and a co-NP problem.

Examples: Critical 3-SAT, **Critical 3-Colorability**

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 (Fagin, K …, Popa - 2003):

Let **M** = (**S**, **T**, Σ_{st}, Σ_t) be 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.

Greedy Algorithm for Computing the Core

M = (**S**, **T**, Σ_{st}, Σ_t) such that Σ_{st} are s-t tgds and Σ_t are target egds

Algorithm Greedy

Input: Source instance I

Output: The core of the universal solutions for I, if solutions exist; "failure", if no solutions exist.

1. Chase I with Σ_{st} to produce a pre-universal solution J for I.
2. Chase J with Σ_t; if the chase fails, return "failure"; otherwise, let J′ be the canonical universal solution produced by the chase.
3. Initialize J* to J′.
4. While there is a fact R(t) in J* such that (I, J* - {R(t)}) $\models$ Σ_{st}, put J* := J* - {R(t)}.
5. Return J* .

Computing the Core

Theorem (Gottlob – PODS 2005):

Let **M** = (**S**, **T**, Σ_{st}, Σ_t) be a schema mapping.

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

Theorem (Gottlob & Nash):

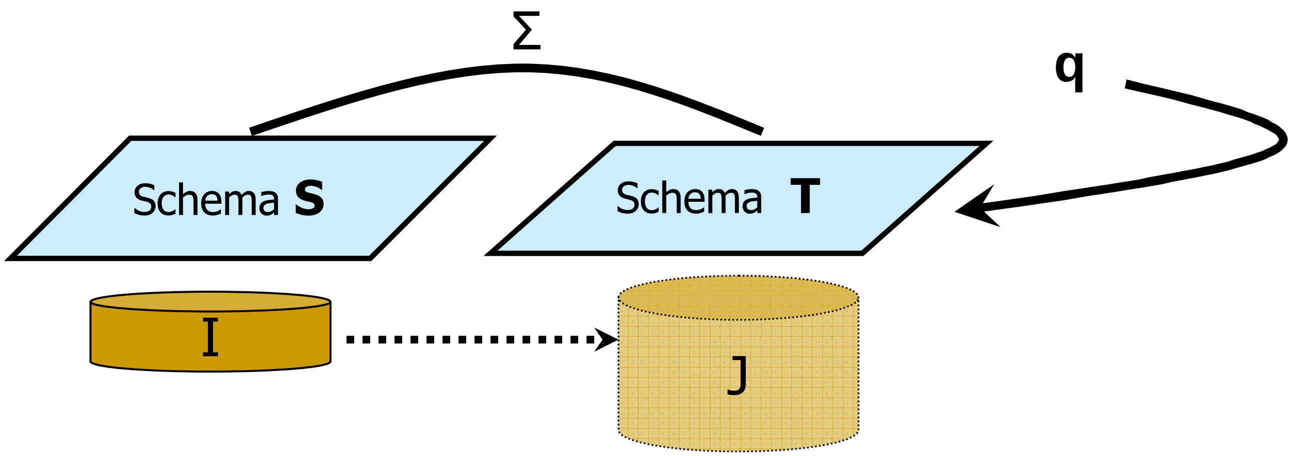
Let **M** = (**S**, **T**, Σ_{st}, Σ_t) be a schema mapping.

If Σ_t is the union of a weakly acyclic set of target tgds with a set of target egds, then the core is polynomial-time computable.

Course Outline – Progress Report

- ✓ Schema Mappings and Data Exchange: Overview
- ✓ Conjunctive Queries and Homomorphisms
- ✓ Data Exchange with Schema Mappings Specified by Tgds and Egds
- ✓ Solutions in Data Exchange
  - Universal Solutions
  - Universal Solutions via the Chase
  - The Core of the Universal Solutions
- Query Answering in Data Exchange

Query Answering in Data Exchange



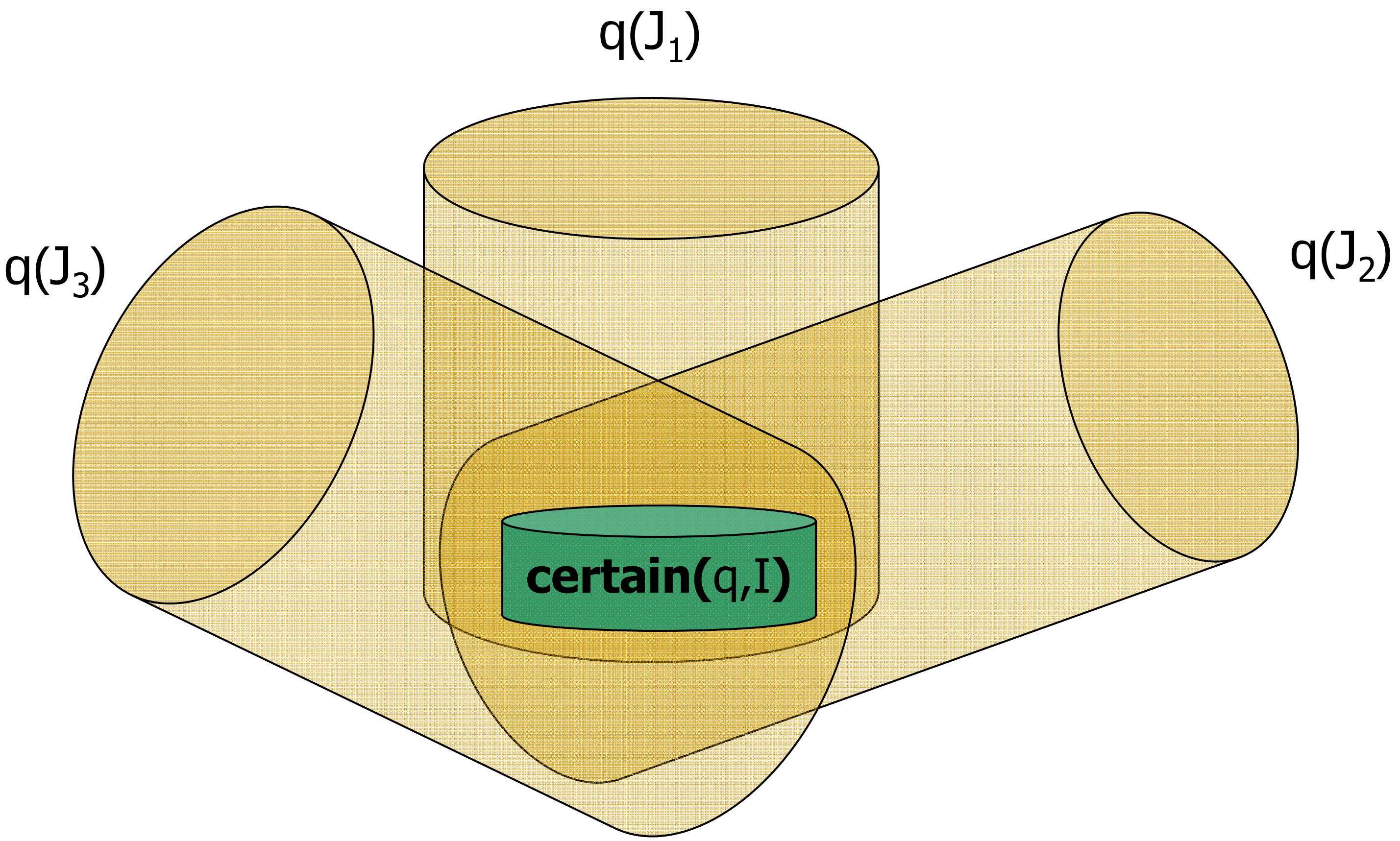
Question: What is the semantics of target query answering?

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



certain(q,I) = $\bigcap$ { q(J): J 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**.

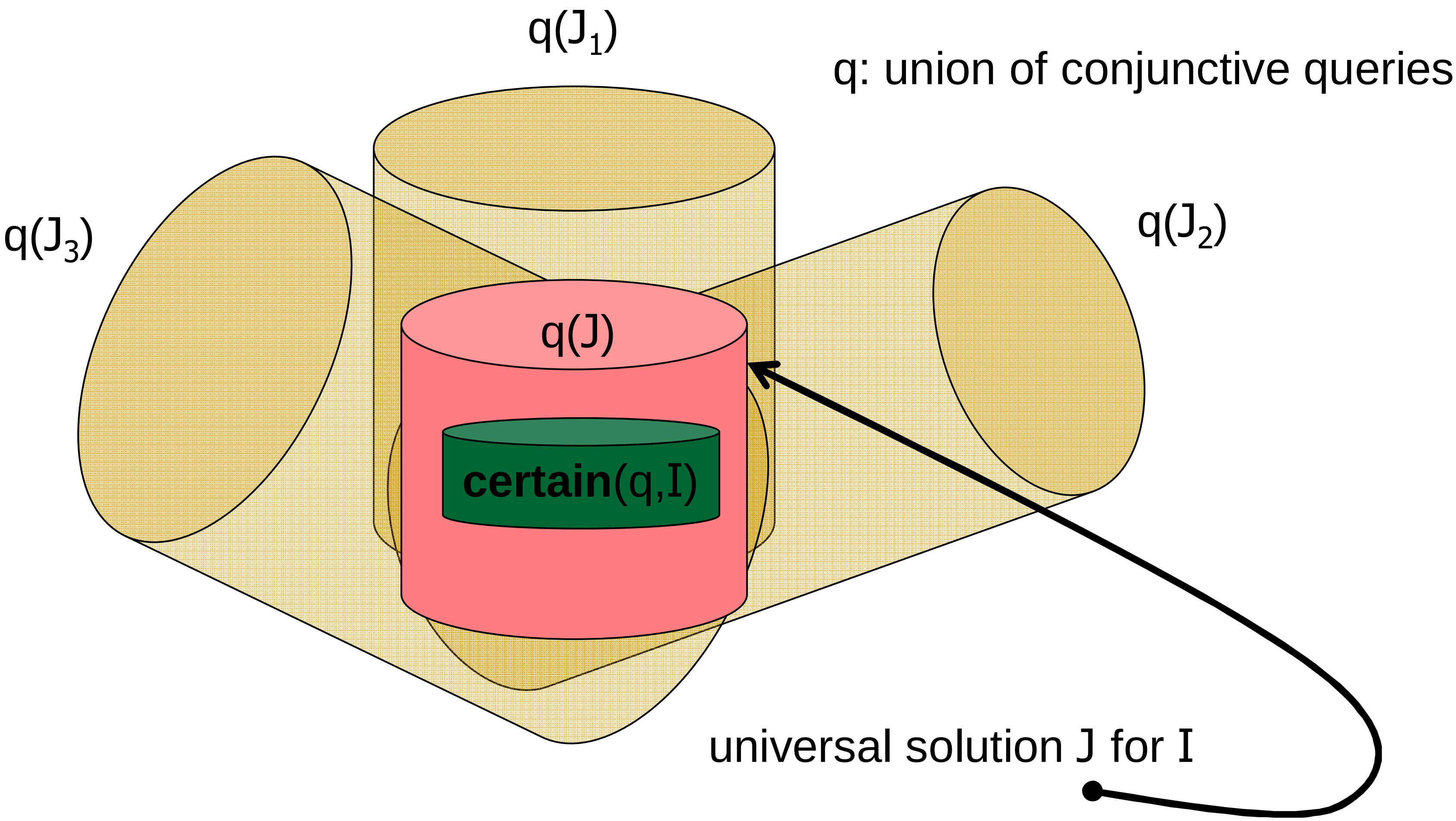
- If I is a source instance and J is a universal solution for I, then

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:

- **certain**(q,I) ⊆ **u-certain**(q,I)
- **certain**(q,I) = **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**.

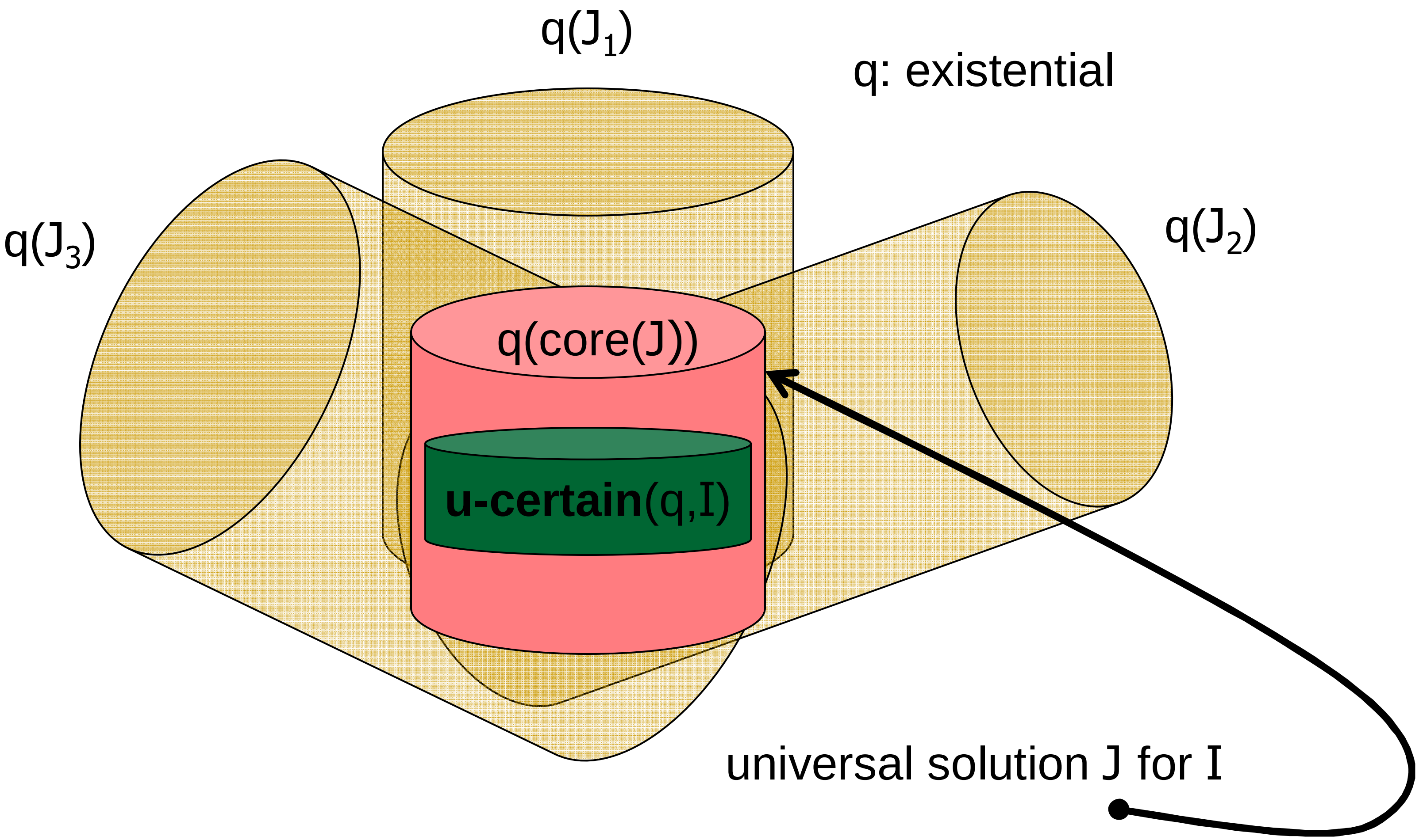
- If I is a source instance and J is a universal solution for I, then

  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)).

Course Outline – Progress Report

- ✓ Schema Mappings and Data Exchange: Overview
- ✓ Conjunctive Queries and Homomorphisms
- ✓ Data Exchange with Schema Mappings Specified by Tgds and Egds
- ✓ Solutions in Data Exchange
  - Universal Solutions
  - Universal Solutions via the Chase
  - The Core of the Universal Solutions
- ✓ Query Answering in Data Exchange

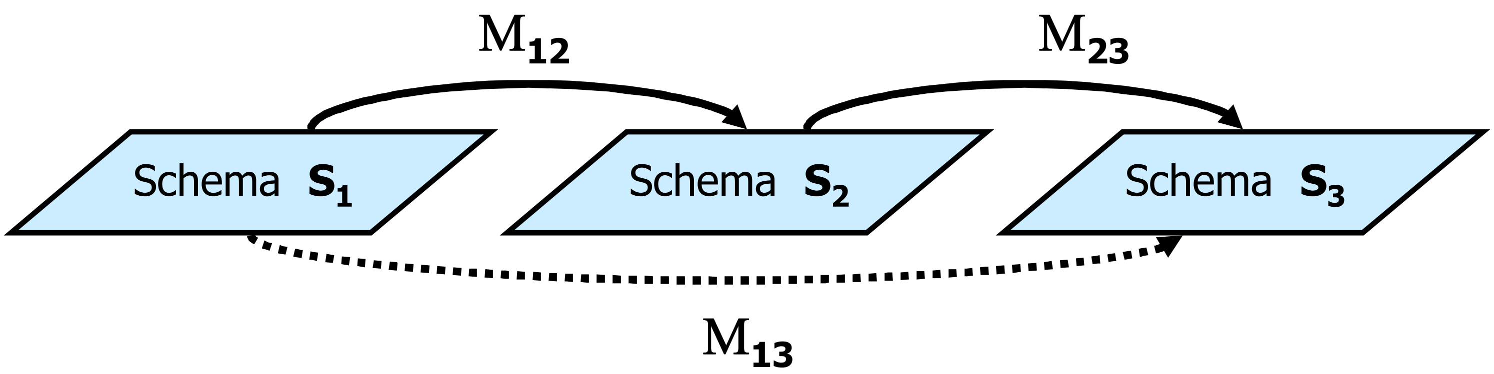
Course Outline – Remaining Topics

- Bernstein's Model Management Framework and Operations on Schema Mappings
- Composing Schema Mappings
- Inverting Schema Mapping
- Extensions of the Framework: Peer Data Exchange
- Open Problems and Research Directions

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
  - Match 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*:*

  Inst(**M**) = { <I,J> | < I,J > ⊨ Σ }.

- **Definition:** (FKPT)

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

  Inst($\mathbf{M}_{13}$) = Inst($\mathbf{M}_{12}$) ° Inst($\mathbf{M}_{23}$), 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

Lower Bounds for Composition

- 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} :
  $\forall x \forall y \forall u \forall v\ (C(x,u) \wedge C(y,v) \wedge F(x,y) \rightarrow D(u,v))$

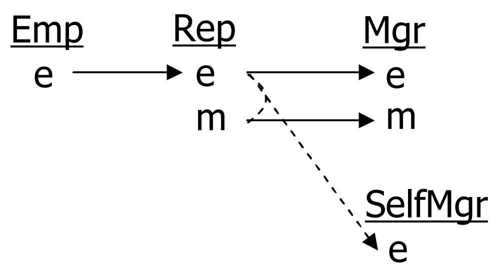
- Given graph **G**=(V, E):
  - Let I_1 = E
  - Let I_3 = { (r,g), (g,r), (b,r), (r,b), (g,b), (b,g) }

Fact:
G is 3-colorable iff $\langle I_1, I_3 \rangle \in$ Inst(M_{12}) ° Inst(M_{23})

- **Theorem (Dawar – 1998):**
  3-Colorability is **not** expressible in $L^{\omega}_{\infty\omega}$

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

Composing SO-Tgds and Data Exchange

Theorem (FKPT):

- The composition of two SO-tgds is definable by a SO-tgd.

- There is an (exponential-time) 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.

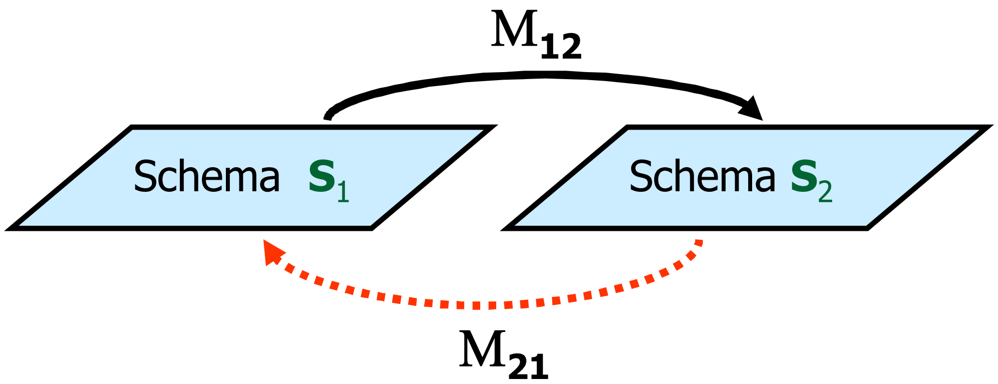
Synopsis of Schema Mapping Composition

- 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 are the ***right*** class for composing s-t tgds: Every SO-tgd defines the composition of finitely many schema mappings, each specified by a finite set of s-t tgds

Related Work on Schema Mappings

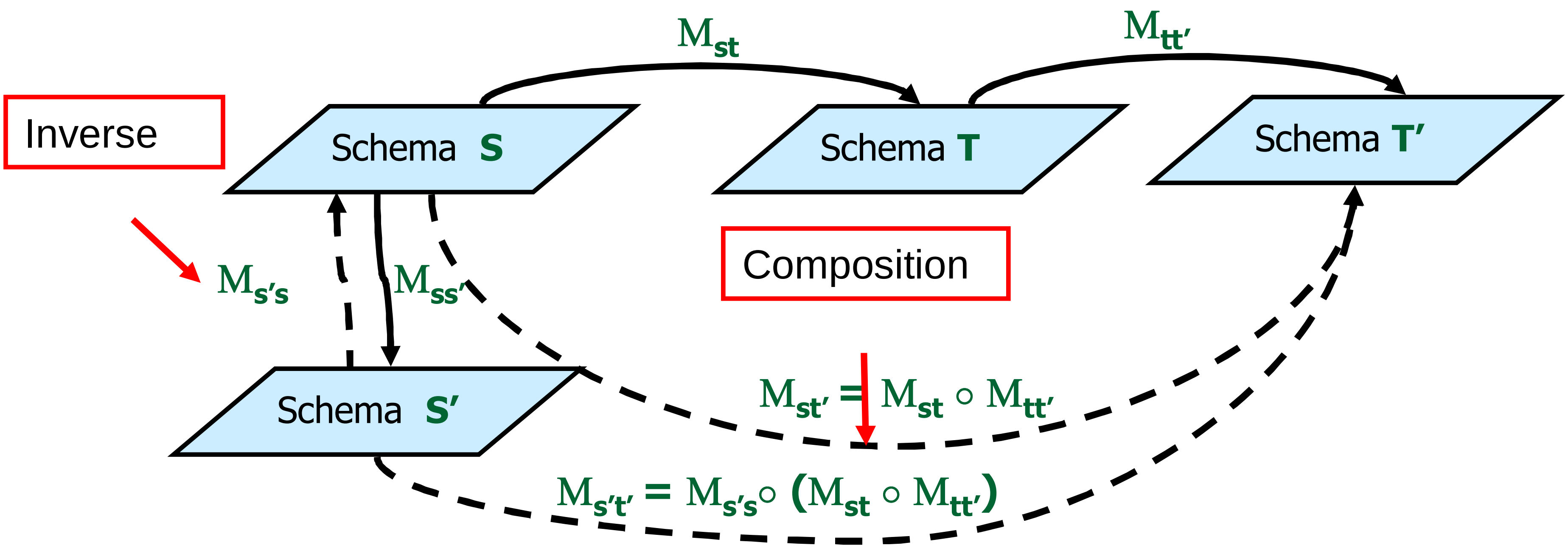
- S. Melnik, Generic Model Management, Ph.D. thesis, 2005
- A. Nash, Ph. Bernstein, S. Melnik (PODS 2005):
  Composition of schema mappings given by source-to-target and target-to-source embedded dependencies
- M. Arenas and L. Libkin (PODS 2005)
  XML Data Exchange
- F. Afrati, C. Li, V. Pavlaki
  Data exchange with s-t tgds containing inequalities

Inverting Schema Mapping



- Given M_{12}, find M_{21} that "undoes" M_{12}
- Inverting schema mappings can be applied to schema evolution

Applications to Schema Evolution



Fact:

Schema Evolution can be analyzed using the composition and the Inverse operators.

Semantics of the Inverse Operator

- Finding the "right" semantics of the inverse operator is a delicate task.
- Naïve approach:
  - If **M** = (**S**, **T**, Σ) is a schema mapping, let
    Inst(**M**) = { (I,J): (I,J $\models \Sigma$ }
  - Define **M*** = (**T**, **S**, Σ^*) to be an inverse of M if
    Inst(**M***) = { (J,I): (I,J) $\models \Sigma$ }
  - This does **not** work if Σ, Σ^* are sets of tgds:

    The reason is that, for schema mappings specified by tgds,
    if (I,J) $\in$ Inst(**M**), I′ $\subseteq$ I, J$\subseteq$ J′, then (I′,J′) $\in$ Inst(**M**).
    However, { (J,I): (I,J) $\models \Sigma$ } does **not** have this property.

Semantics of the Inverse Operator

Fagin – PODS 2006

- **Motivation:** an **inverse** of a function f is a function f′ s.t.

  $$f \circ f' = id,$$

  where id is the **identity function** f(x)=x

- **Key Idea:**
  - Define first the **identity schema mapping Id**
  - Call a schema mapping **M′** an **inverse** of **M** if

    $$\mathbf{M} \circ \mathbf{M}' = \mathbf{Id}$$

The Identity Schema Mapping

Definition: Let **S** be a schema.

For each relation symbol R in **S**, let R* be a replica of R.

Let $\mathbf{S^*} = \{ R^*: R \in \mathbf{S} \}$.

The **identity schema mapping on S** is the schema mapping

$$\mathbf{Id_S} = (\mathbf{S}, \mathbf{S^*}, \Sigma_{Id}(\mathbf{S}))$$

where $\Sigma_{Id}(\mathbf{S})$ consists of the dependencies

$$R(x) \rightarrow R^*(x),$$

for every relation symbol $R \in \mathbf{S}$.

Inverting Schema Mapping

Definition: Fagin – 2006

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

A schema mapping **M*** = (**T**, **S***, Σ*) is an **inverse** of **M** if

$$\mathbf{M} \circ \mathbf{M^*} = \mathbf{Id_S}$$

Example:

An inverse of the identity mapping

$$\mathbf{Id_S} = (\mathbf{S}, \mathbf{S^*}, \Sigma_{Id}(\mathbf{S})) \text{ on } \mathbf{S}$$

is the identity mapping

$$\mathbf{Id_{S^*}} = (\mathbf{S^*}, \mathbf{S^{**}}, \Sigma_{Id}(\mathbf{S^*})) \text{ on } \mathbf{S^*}.$$

Inverses of Schema Mappings

Example: Let **M** be the schema mapping specified by the tgd

$$P(x) \rightarrow Q(x,x).$$

Then:

- The schema mapping **M′** specified by the tgd

  $$Q(x,y) \rightarrow P^*(x)$$

  is an inverse of **M**.

- The schema mapping **M′′** specified by the tgd

  $$Q(x,y) \rightarrow P^*(y)$$

  is also an inverse of **M**.

Conclusion:

Inverses need not be unique up to logical equivalence.

The Unique Solutions Property

Theorem: Fagin – 2006

If a schema mapping **M** has an inverse, then **M** must have the **unique-solutions property**:

If I_1 and I_2 are source instances such that $I_1 \neq I_2$,

then **Sol**(**M**, I_1) $\neq$ **Sol**(**M**, I_2).

Note:

- The unique-solutions property is a necessary condition for invertibility.
- Hence, it can be used a sufficient condition for non-invertibility.

Non-invertible Schema Mappings

Fact: None of the following schema mappings is invertible, as none satisfies the unique-solutions property:

- **Projection:**

  $P(x,y) \rightarrow Q(y)$

- **Union:**

  $P(x) \rightarrow Q(x)$

  $R(x) \rightarrow Q(x)$

- **Decomposition:**

  $P(x,y,z) \rightarrow Q(x,y) \wedge T(y,z)$

Inverting Schema Mappings

Good News:

Rigorous semantics of the inverse operator has been given.

Not-so-good News:

It is a rare that a schema mapping has an inverse, so the applicability of the inverse operator is limited

Ongoing work: (FKPT)

Quasi-inverses of schema mappings,

a relaxation of the notion of inverses of schema mapping.

Course Outline – Remaining Topics

- ✓ Bernstein's Model Management Framework and Operations on Schema Mappings
- ✓ Composing Schema Mappings
- ✓ Inverting Schema Mapping
- Extensions of the Framework: Peer Data Exchange
- Open Problems and Research Directions

Extending the Data Exchange Framework

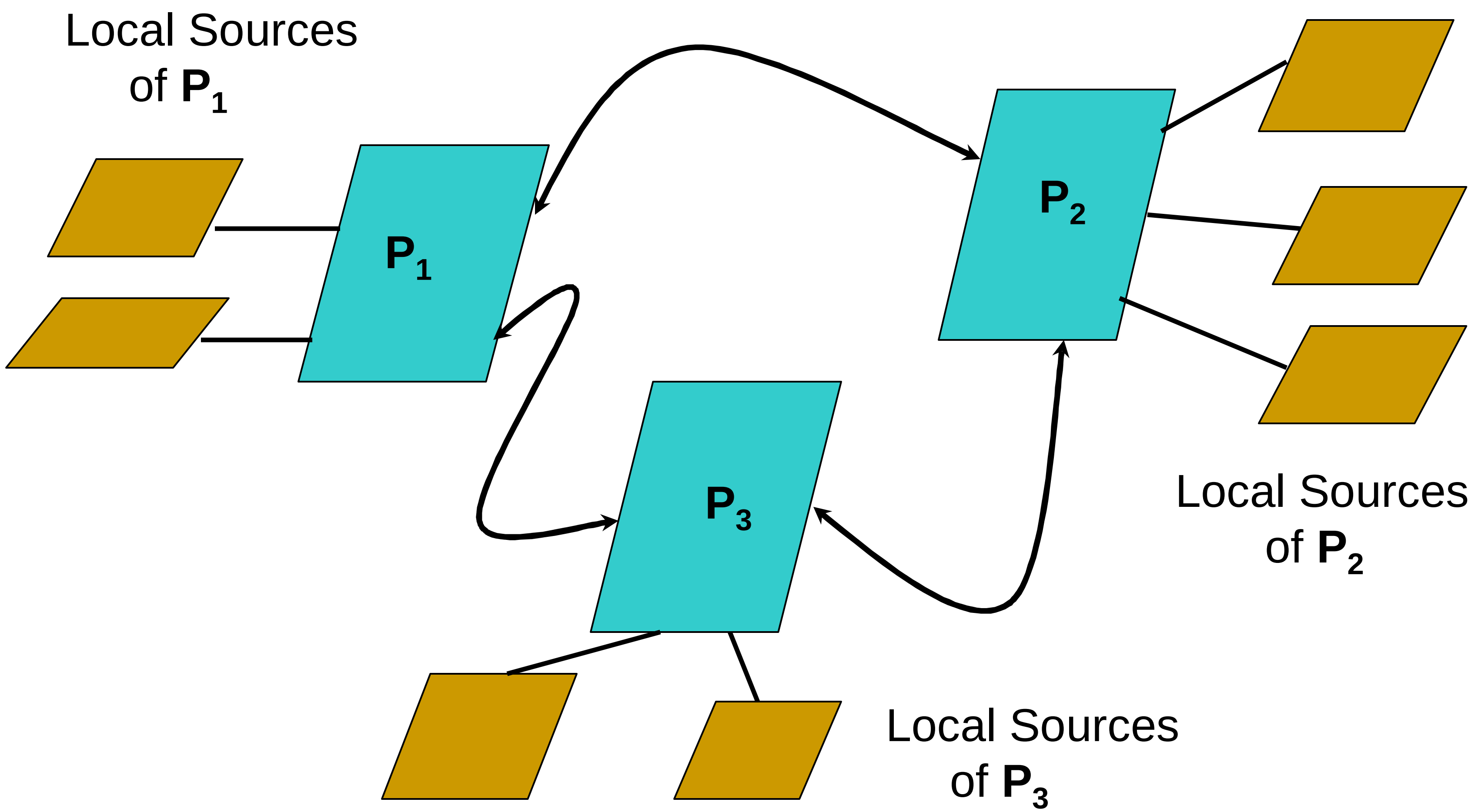
- The original data exchange formulation models a situation in which the target is a **passive receiver** of data from the source:
  - The constraints are “**directed**” from the source to the target.
  - Data is moved from the source to the target only; moreover, originally the target has no data.
- It is natural to consider **extensions** to this framework:
  - Bidirectional constraints between source and target
  - Bidirectional movement of data from the source to the target and from an already populated target to the source.

Peer Data Management Systems (PDMS)

- Halevy, Ives, Suciu, Tatarinov – ICDE 2003
- Motivated from building the Piazza data sharing system
- Decentralized data management architecture:
  - Network of peers.
  - Each peer has its own schema; it can be a mediated global schema over a set of local, proprietary sources.
  - Schema mappings between sets of peers with constraints:
    - $q_1(\mathbf{A}_1) = q_2(\mathbf{A}_2)$
    - $q_1(\mathbf{A}_1) \subseteq q_2(\mathbf{A}_2)$,

    where $q_1(\mathbf{A}_1)$, $q_2(\mathbf{A}_2)$ are conjunctive queries over sets of schemas.

Peer Data Management Systems



Peer Data Management Systems

- **Theorem (HIST03):** There is a PDMS **P*** such that:
  - The existence-of-solutions problem for **P*** is undecidable.
  - Computing the certain answers of conjunctive queries is an undecidable problem.

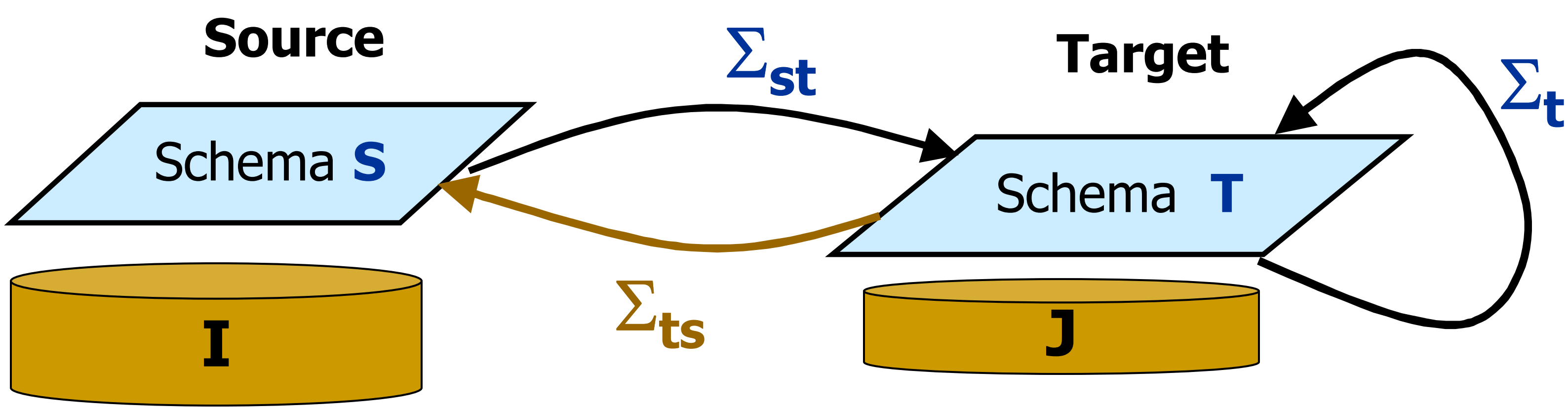
- **Moral:**
  - Expressive power comes at a high cost.
  - To maintain decidability, we need to consider extensions of data exchange that are less powerful than arbitrary PDMS.

Peer Data Exchange (PDE)

- Fuxman, K …, Miller, Tan - PODS 2005
- Peer Data Exchange models data exchange between two peers that have different roles:
  - The source peer is an **authoritative** source peer.
  - The target peer is willing to accept data from the source peer, provided **target-to-source constraints** are satisfied, in addition to source-to-target constraints.
  - Source data are moved and **added to existing data on the target.**
  - The source data, however, remain **unaltered** after the exchange.

Peer Data Exchange

d3



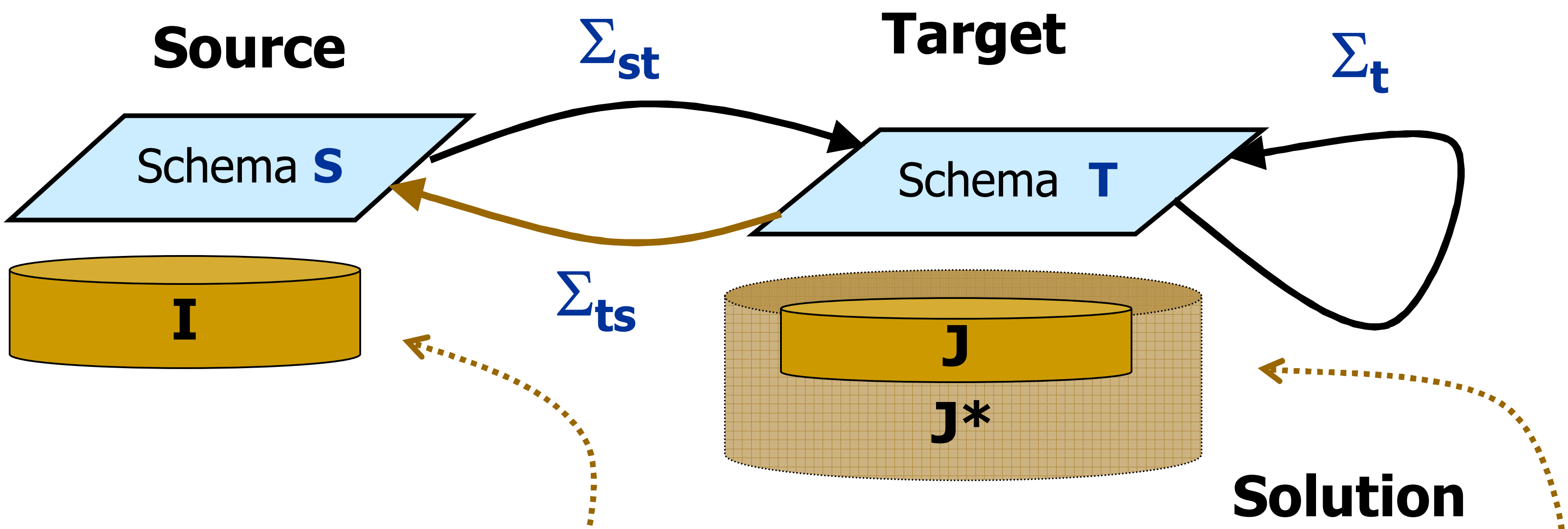
- **Constraints:**
  - $\Sigma_{st:}$ source-to-target tgds, Σ_t target tgds and egds
  - Σ_{ts} target-to-source tgds,
- **Extensions to Data Exchange:**
  - Target-to-source dependencies
  - Input target instance

d3

Modeling “authority” relationships
Asymmetry between source and target: source cannot be modified by \Sigma_{ts}
db2admin, 5/22/2005

Solutions in Peer Data Exchange

d4



- A solution for (I,J) is a target instance J* such that:
  1. **J $\subseteq$ J***
  2. $\langle I, J^* \rangle \models \Sigma_{st}$
  3. $J \models \Sigma_t$
  4. **$\langle J^*, I \rangle \models \Sigma_{ts}$**

Asymmetry models the authority of the source

d4 Modeling “authority” relationships
Asymmetry between source and target: source cannot be modified by \Sigma_{ts}
db2admin, 5/22/2005

Algorithmic Problems in PDE

- **Definition:** Peer Data Exchange **P** = (S,T, Σ_{st}, Σ_t, Σ_{ts})

  The **existence-of-solutions problem** **Sol(P):**

  Given a source instance I and a target instance J, is there a solution J* for (I,J) in **P**?

- **Definition:** Peer Data Exchange **P** = (S,T, Σ_{st}, Σ_t, Σ_{ts}), query q

  Computing the certain answers of q with respect to P:

  Given a source instance I and a target instance J, compute

$$\textbf{certain}_{\mathbf{P}}(q,(I,J)) = \bigcap \{q(J^*): J^* \text{ is a solution for } (I,J)\}$$

Results for Peer Data Exchange: Overview

- **Upper Bounds:** For every PDE **P** = $(S, T, \Sigma_{st}, \Sigma_t, \Sigma_{ts})$ with Σ_t weakly acyclic set of tgds and egds, and every target conjunctive query q:
  - **Sol(P)** is in NP.
  - **certain$_P$**(q,(I,J)) is in coNP.

- **Lower Bounds:** There is a PDE **P** = $(S, T, \Sigma_{st}, \Sigma_t, \Sigma_{ts})$ with $\Sigma_t = \emptyset$ and a target conjuctive query q such that:
  - **Sol(P)** is NP-complete.
  - **certain$_P$**(q,(I,J)) is coNP-complete.

- **Tractability Results:**
  - Syntactic conditions on PDE settings and on conjunctive queries that guarantee tractability of **Sol(P)** and of **certain$_P$**(q,(I,J)).

Upper Bounds

Theorem: Let **P** = (S,T, Σ_{st}, Σ_t, Σ_{ts}) be a PDE setting such that Σ_t is the union of a weakly acyclic set of tgds with a set of egds.
Then:

- **Sol(P)** is in NP.
- **certain$_P$**(q,(I,J)) is in coNP, for every monotone target query q.

Hint of Proof: Establish a ***small model property:***

- Whenever a solution J′ exists, a **"small"** solution J* must exist
  "small" = polynomially-bounded by the size of I and J

Solution-aware chase

- Instead of creating null values, use values from the given solution J′ to witness the existentially-quantified variables.
- The result of the solution-aware chase of (I,J) with $\Sigma_{st} \cup \Sigma_t$ and the given solution J′ is a **"small"** solution J*.

a2

Lower Bounds

Theorem: There is a PDE setting **P** = (S,T, Σ_{st}, Σ_t, Σ_{ts}) with $\Sigma_t = \emptyset$ and a target conjuctive query q such that:

- **Sol(P)** is NP-complete.
- **certain$_P$**(q,(I,J)) is coNP-complete.

Proof: Reduction from the 3-COLORABILITY Problem

- S = {D, E} binary symbols, T = {C, F} binary symbols

Σ_{st}: $E(x,y) \rightarrow \exists\, u C(x,u)$

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

Σ_{ts}: $C(x,u) \wedge C(y,v) \wedge F(x,u) \rightarrow D(u,v)$

- Source instance: D = { (r,g), (g,r), (b,r), (r,b), (g,b), (b,g) }

  E = edge relation of a graph.

a2 say that we give an alternative proof using a reduction from the CLIQUE problem.... use this reduction to show the tightness of the tractable class

afuxman, 6/7/2005

Comparison of Complexity Results

	SOL(P)	**Certain_P(q,(I,J))**
Data Exchange (FKMP03)	PTIME; trivial, if $\Sigma_t = \emptyset$.	PTIME
Peer Data Exchange	in NP; can be NP-complete, even if $\Sigma_t = \emptyset$.	in coNP; can be coNP-complete, even if $\Sigma_t = \emptyset$.
PDMS (HIST03)	can be undecidable.	can be undecidable.

Tractable Peer Data Exchange

- **Goal:** Identify **syntactic conditions** on the dependencies of peer data exchange settings **P** that guarantee polynomial-time algorithms for **Sol(P)**.

- **Key concepts**: **marked positions** and **marked variables**

  - Σ_{st}: $D(x,y) \rightarrow \exists z \, \exists w \, P(x,z,y,w)$

    2nd and 4th position of P are **marked**

  - Σ_{ts}: $P(x,u,y,v) \rightarrow E(u,v)$

    u and v are **marked variables**

Tractable Peer Data Exchange Settings

Definition: $\mathbf{C}_{tract}$ is the class of all PDE **P** = $(S,T, \Sigma_{st}, \Sigma_t, \Sigma_{ts})$ with $\Sigma_t = \emptyset$ and such that the marked variables obey certain syntactic conditions, including:

> if two marked variables appear together in an atom in the RHS of a dependency in Σ_{ts}, then they must appear together in an atom in the LHS of that dependency - or not appear at all.

Note: Consider the PDE setting **P** = $(S,T, \Sigma_{st}, \Sigma_t, \Sigma_{ts})$ with

$$\Sigma_{st}: \quad E(x,y) \rightarrow \exists\, u C(x,u)$$
$$E(x,y) \rightarrow F(x,y)$$

$$\Sigma_{ts}: \quad C(x,u) \wedge C(y,v) \wedge F(x,u) \rightarrow D(u,v)$$

P is not in $\mathbf{C}_{tract}$ because the marked variables z and z′ **violate** the above syntactic condition.

Practical Subclasses of $\mathbf{C}_{\text{tract}}$

- **Full source-to-target dependencies**

  $$\phi_s(\mathbf{x},\mathbf{x'}) \rightarrow \psi_t(\mathbf{x})$$

- Arbitrary target-to-source dependencies

- Arbitrary source-to-target dependencies
- **Local-as-view target-to-source dependencies**

  $$R(\mathbf{x}) \rightarrow \exists\ \mathbf{y}\ \beta(\mathbf{x},\mathbf{y})$$

Existence of Solutions in $\mathbf{C}_{\text{tract}}$

Theorem: If **P** is a peer data exchange setting in $\mathbf{C}_{\text{tract}}$, then the existence-of-solutions problem **Sol(P)** is in PTIME.

Proof Ingredients:

- Solution-aware chase.
- Homomorphism techniques.

Maximality of $\mathbf{C}_{\text{tract}}$

Fact: $\mathbf{C}_{\text{tract}}$ is a **maximal** tractable class:

- Minimal relaxations of the conditions of $\mathbf{C}_{\text{tract}}$ can lead to intractability (**Sol(P)** becomes NP-hard).

- The intractability boundary is also crossed if

  Σ_{st} and Σ_{ts} satisfy the conditions of $\mathbf{C}_{\text{tract}}$, but
  - there is a single egd in the target;

  or,
  - there is a single full tgd in the target.

Query Answering in $\mathbf{C}_{tract}$

Theorem: There is a PDE setting **P** in $\mathbf{C}_{tract}$ and a target conjunctive query q such that $\mathbf{certain_P}$(q,(I,J)) is coNP-complete.

Theorem: If **P** is a PDE setting in $\mathbf{C}_{tract}$ and q is a target conjunctive query such that each marked variable occurs only once in q, then $\mathbf{certain_P}$(q,(I,J)) is in PTIME.

Corollary: If **P** is a PDE setting such that Σ_{st} is a set of full tgds and $\Sigma_t = \emptyset$, then $\mathbf{certain_P}$(q,(I,J)) is in PTIME for every target conjunctive query q.

Universal Bases in Peer Data Exchange

Fact: In peer data exchange, universal solutions need **not** exist (even if solutions exist).

Substitute: Universal basis of solutions

Definition: PDE **P** = $(S,T, \Sigma_{st}, \Sigma_t, \Sigma_{ts})$

A **universal basis for** (I,J) is a set **U** of solutions for (I,J) such that for every solution J*, there is a solution J_u in **U** such that a homomorphism from J_u to J* exists.

Universal Bases in Peer Data Exchange

Theorem: For **P** = (S,T, Σ_{st}, Σ_t, Σ_{ts}) with $\Sigma_t = \emptyset$:

- A solution exists if and only if a **universal basis** exists.
- There is an exponential-time algorithm for constructing a universal basis, when a solution exists.
- Every universal basis may be of exponential size

  (even for PDEs in $\mathbf{C}_{tract}$).

Synopsis

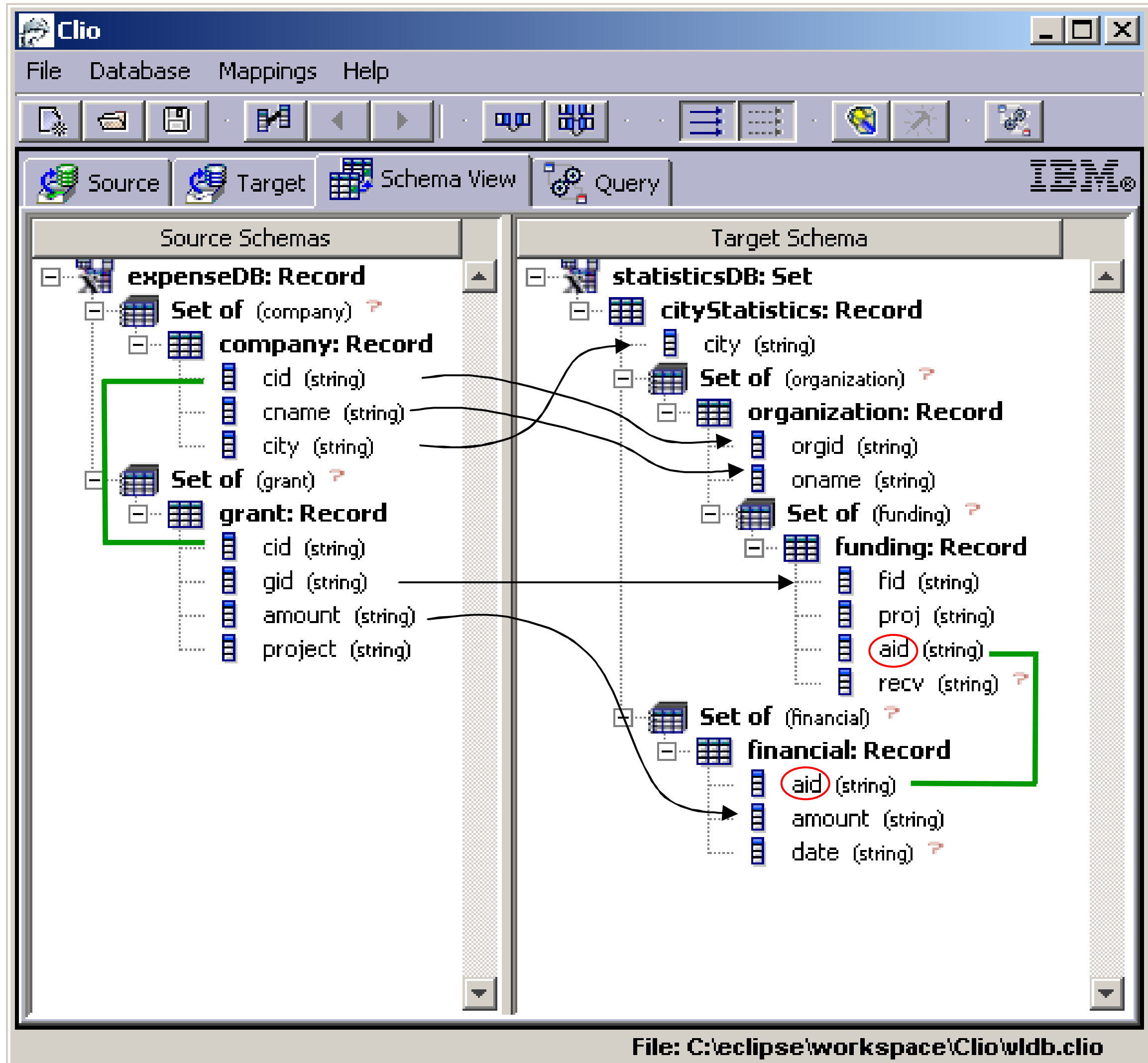
- Peer Data Exchange is a framework that:
  - generalizes Data Exchange;
  - is a special case of Peer Data Management Systems.
- This is reflected in the complexity of testing for solutions and computing the certain answers of target queries.
- We identified a "**maximal**" class of Peer Data Exchange settings for which **Sol(P)** is in PTIME.
- Much more remains to be done to delineate the boundary of tractability and intractability in Peer Data Exchange.

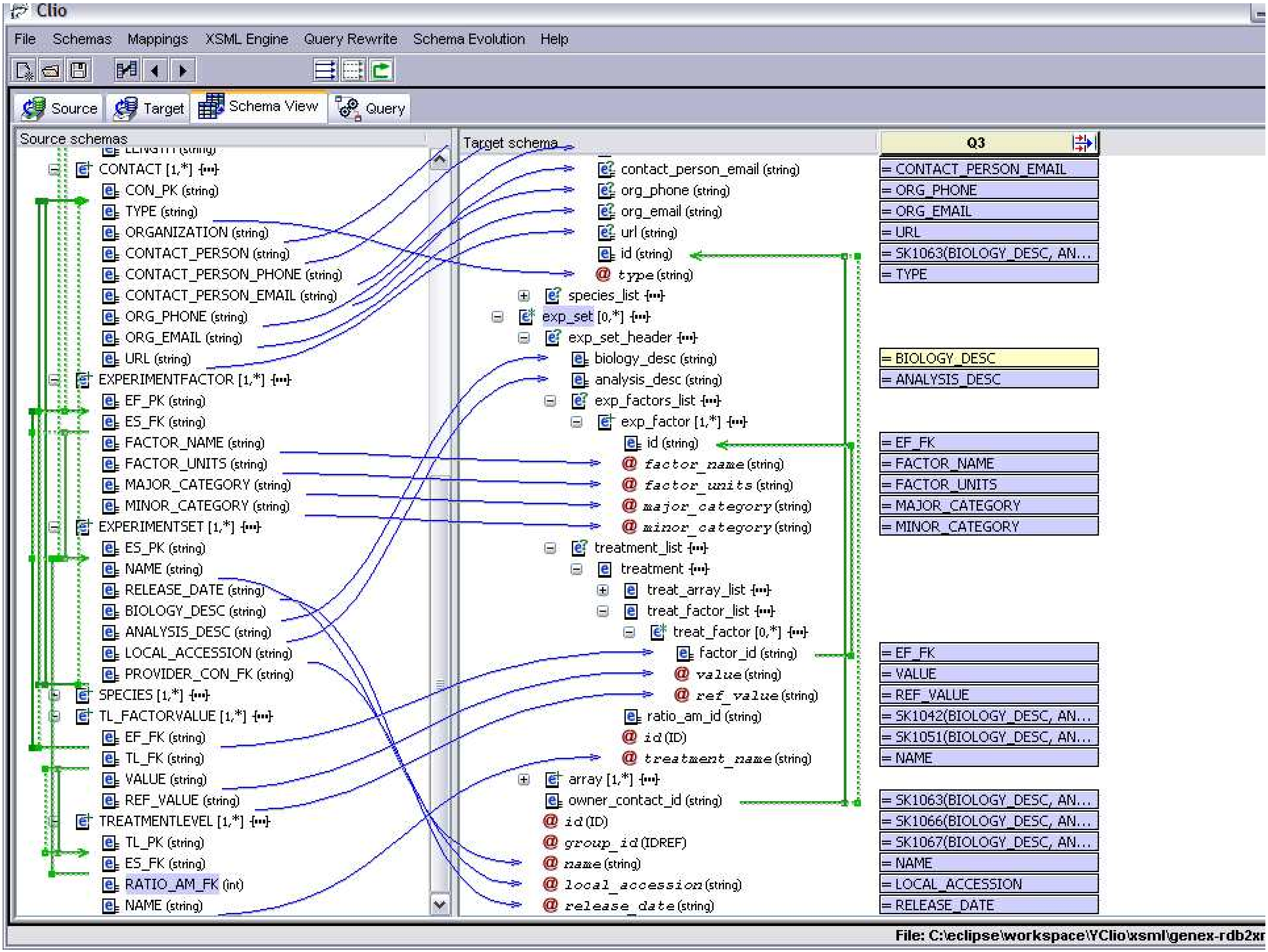
Theory and Practice

- 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 IBM products (IBM Information Server).

Some Features of Clio

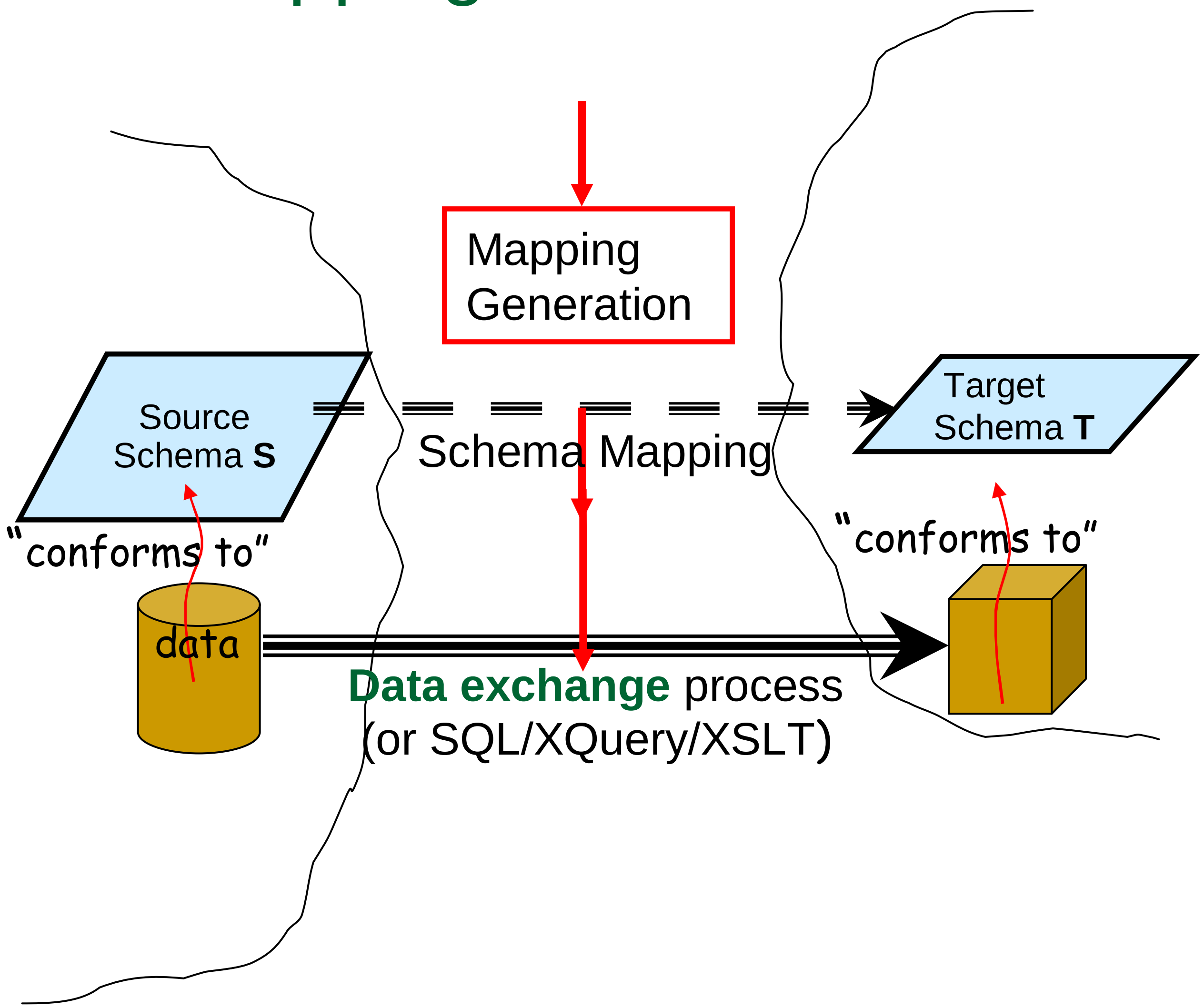
- Supports nested structures
  - Nested Relational Model
  - Nested Constraints
- Automatic & semi-automatic discovery of attribute correspondence.
- Interactive derivation of schema mappings.
- Performs data exchange





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



Open Problems and Directions for Research

- Investigate further the inverse operator and its variants.

- Develop rigorous semantics for the other operators in Bernstein's framework.

- Develop a theory of schema mapping optimization:
  identify the key parameters and appropriate "optimization" functions that will allow us to compare schema mappings and design algorithms for optimizing them.

- Unify data integration and data exchange:
  Develop flexible information integration systems that support both mediation and materialization.

Pasteur's Quadrant

	Consideration of use? No	Consideration of use? Yes
Quest for fundamental understanding? Yes	Pure Basic Research (Bohr)	Use-inspired basic research (Pasteur)
Quest for fundamental understanding? No		(Pure) applied research (Edison)

Stokes, Donald E., *Pasteur's Quadrant: Basic Science and Technological Innovation,* 1997, *Figure 3.5*