Logic, Data, and Incomplete Information

Phokion G. Kolaitis

UC Santa Cruz & IBM Research

Lecture 4

Query Evaluation on a Collection of Databases

Note:

- There are many natural scenarios, however, in which the query evaluation problem involves a query and a collection of databases.
- Typically, this collection of databases are the "completions" of some "incomplete" database.
- These "completions" can be thought of as the possible worlds arising from an "incomplete" database.

Question: What is the semantics of a query posed on a collection of databases?

Certain Answers

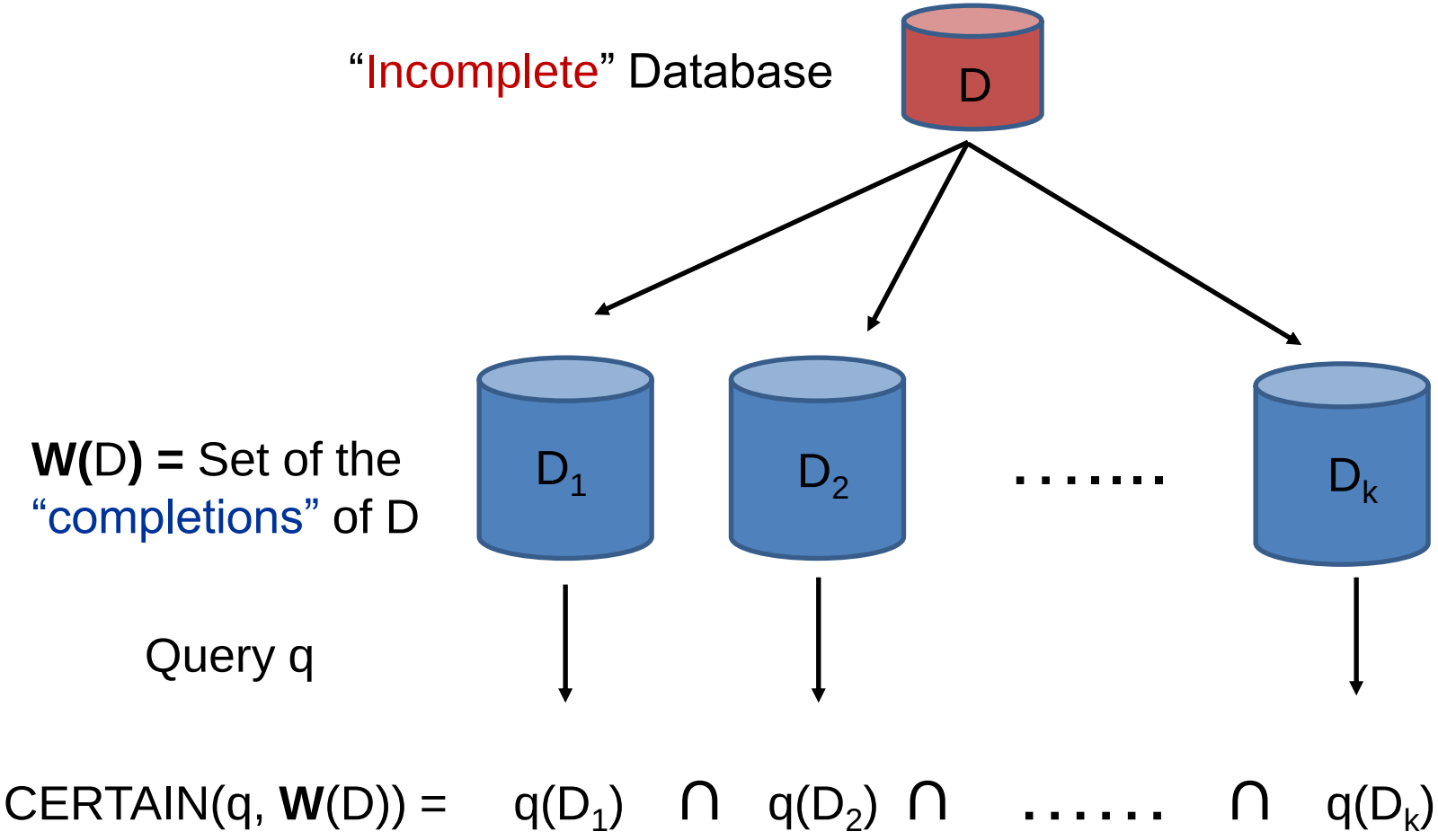
Definition:

- Let q be a k-ary query and ***W*** a collection of databases.
  The certain answers of q on ***W*** is the set

  CERTAIN(q,***W***) = $\bigcap_{D \text{ in } \boldsymbol{W}} q(D) = \{ \mathbf{a}: \mathbf{a} \in q(D), \text{ for every D in } \boldsymbol{W} \}$

- Let q be a Boolean query and ***W*** a collection of databases.
  The certain answers of q on ***W*** is
  CERTAIN(q, ***W***) = 1, if q(D) = 1 for every D in ***W***;
  CERTAIN(q, ***W***) = 0, if q(D) = 0 for some D in ***W***.

Visualizing the Certain Answers



Possible Answers

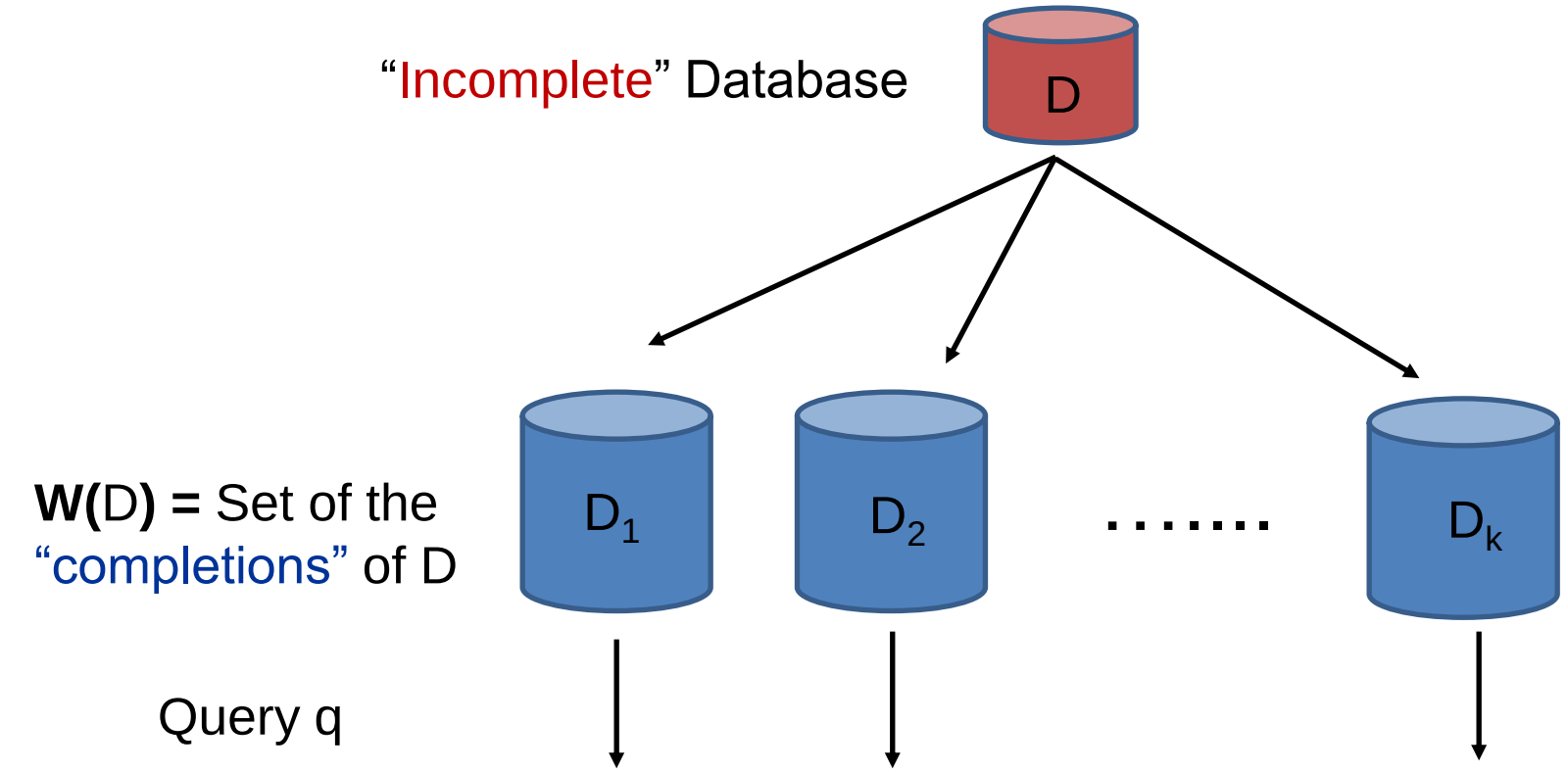
Definition:

- Let q be a k-ary query and ***W*** a collection of databases.
  The possible answers of q on ***W*** is the set

  POSSIBLE(q,***W***) $= \bigcup_{D \text{ in } \boldsymbol{W}} q(D) = \{ \mathbf{a}: \mathbf{a} \in q(D), \text{ for some D in } \boldsymbol{W} \}$

- Let q be a Boolean query and ***W*** a collection of databases.
  The possible answers of q on ***W*** is
  POSSIBLE(q, ***W***) = 1, if q(D) = 1 for some D in ***W***;
  POSSIBLE(q, ***W***) = 0, if q(D) = 0 for every D in ***W***.

Visualizing the Possible Answers



Certain Answers vs. Possible Answers

Note:

- A certain answer a provides a strong guarantee:
  It is an answer to the query on every possible world.

- A possible answer a provides a weaker guarantee:
  It is an answer to the query in at least one possible world.

Inconsistent Databases and Consistent Answers

- Inconsistent Databases
  - Possible Worlds: The repairs of an inconsistent database
  - Consistent answers are the certain answers over the repairs.

- Boolean query q, set Σ of functional dependencies
  CERTAINTY(q, Σ): Given I, is q(J)=1 on every repair J of I?

- CERTAINTY(q, Σ) can be coNP-complete
  "sink" query q: $\exists x, y, z \left(R(x, y) \wedge S(z, y)\right)$, where x and z are keys.

- Dichotomy Theorem for self-join free conjunctive queries and keys

- Dichotomy Conjecture for arbitrary conjunctive queries and keys.

Certain Answers in Different Contexts

- ✓ Inconsistent Databases
  - Possible Worlds: The repairs of an inconsistent database
  - Consistent answers are the certain answers over the repairs.

- Data Exchange
  - Possible Worlds: The solutions of a source instance.

- Probabilistic Databases
  - Possible Worlds: The databases represented by a probabilistic database.

- Computational Social Choice:
  - Possible Worlds: The completions of a set of partial orders representing partial preferences of voters.

The Data Interoperability Challenge

- Data may reside
  - at several different sites
  - in several different formats (relational, XML, …).

- Applications need to access and process all these data.
  IBM, SAP, Oracle, and Microsoft offer competing software system for data interoperability tasks.

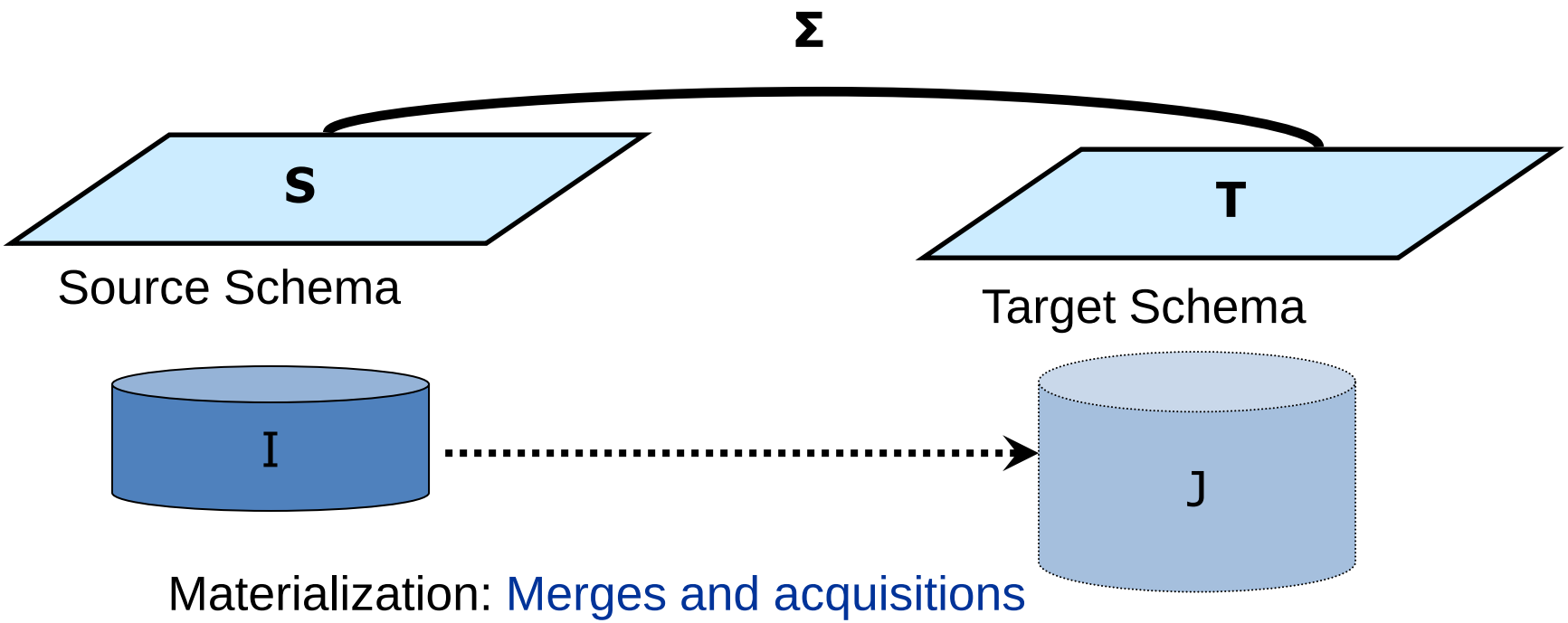
- The research community has formalized and studied different facets of data interoperability, including

  Data Exchange (aka Data Translation)

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

Data Exchange

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



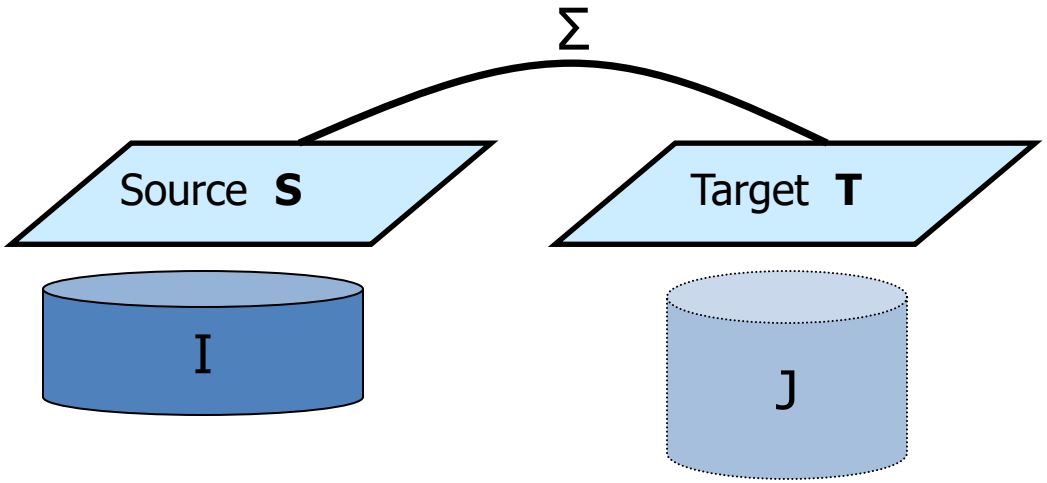
Schema Mappings

- Schema mappings:
  High-level, declarative assertions (constraints) that specify the relationship between two database schemas.

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

- Schema mappings help with the development of tools.

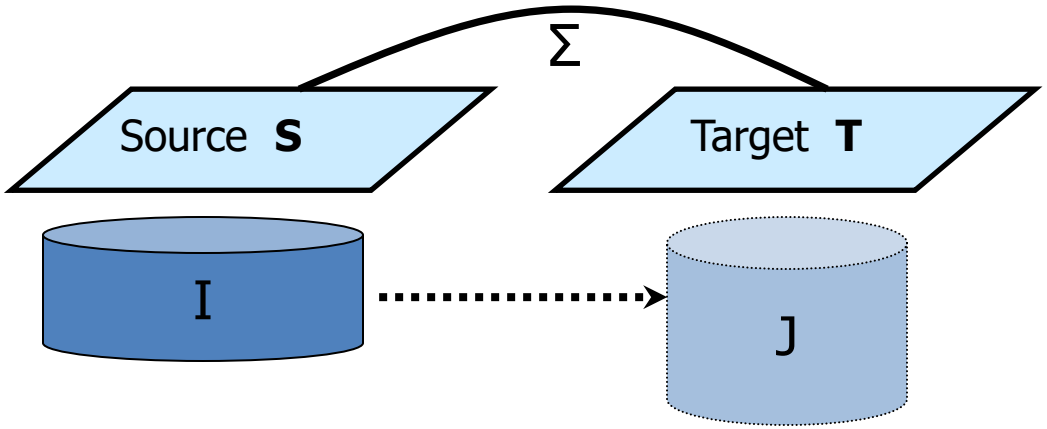
Schema Mappings



- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - A set Σ of high-level, declarative assertions (constraints) that specify the relationship between **S**-instances and **T**-instances.

- Inst(**M**) = { (I, J): I is an **S**-instance, J is a **T**-instance, and (I, J) ⊨ Σ }.

Schema Mappings & Data Exchange



- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - A set Σ of high-level, declarative assertions (constraints) that specify the relationship between **S**-instances and **T**-instances.
- 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 the pair (I, J) satisfies Σ.

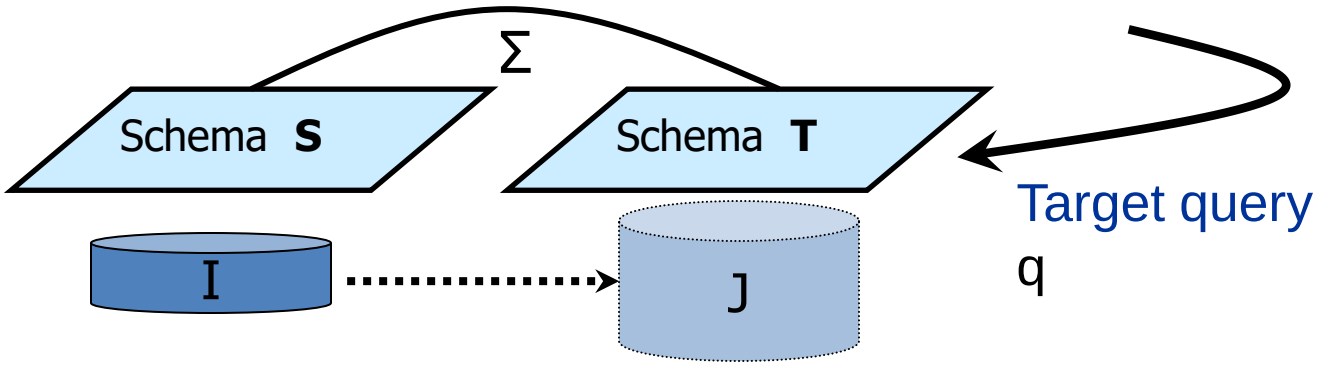
Fact: In general, for a given source instance I,

- No solution for I may exist (the constraints overspecify)

  or

- Multiple solutions for I may exist; in fact, infinitely many solutions for I may exist (the constraints underspecify).

Schema Mappings: Algorithmic Problems



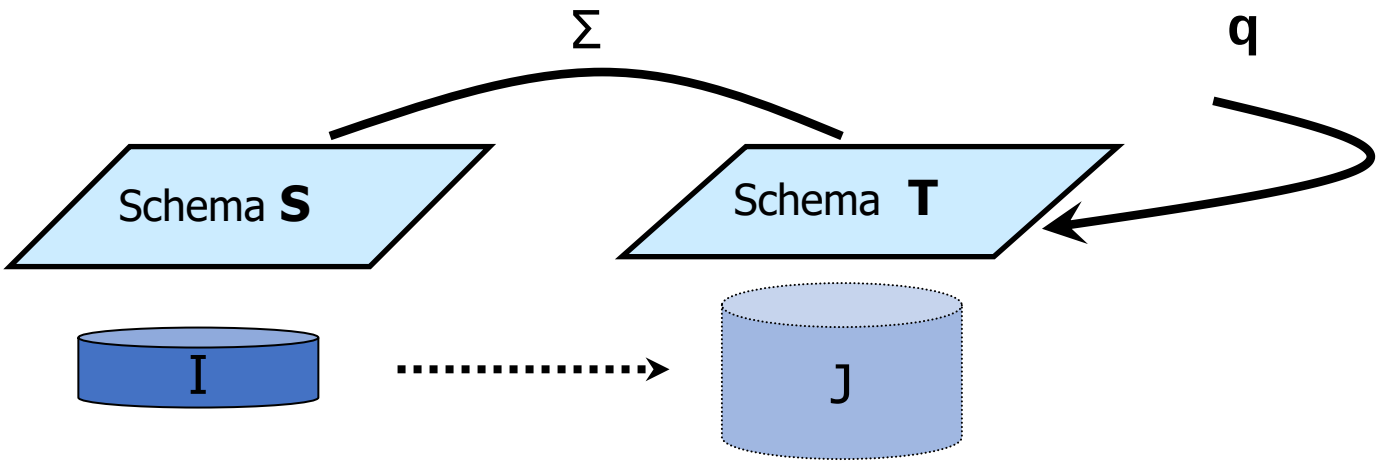
Definition: Schema Mapping **M** = (**S**, **T**, Σ)

- 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, find a solution J for I, provided a solution exists.

- The query answering problem associated with **M** and a target query q:
  Given a source instance I, "evaluate" q on I.

Query Answering in Data Exchange



Question: What is the semantics of target query answering?

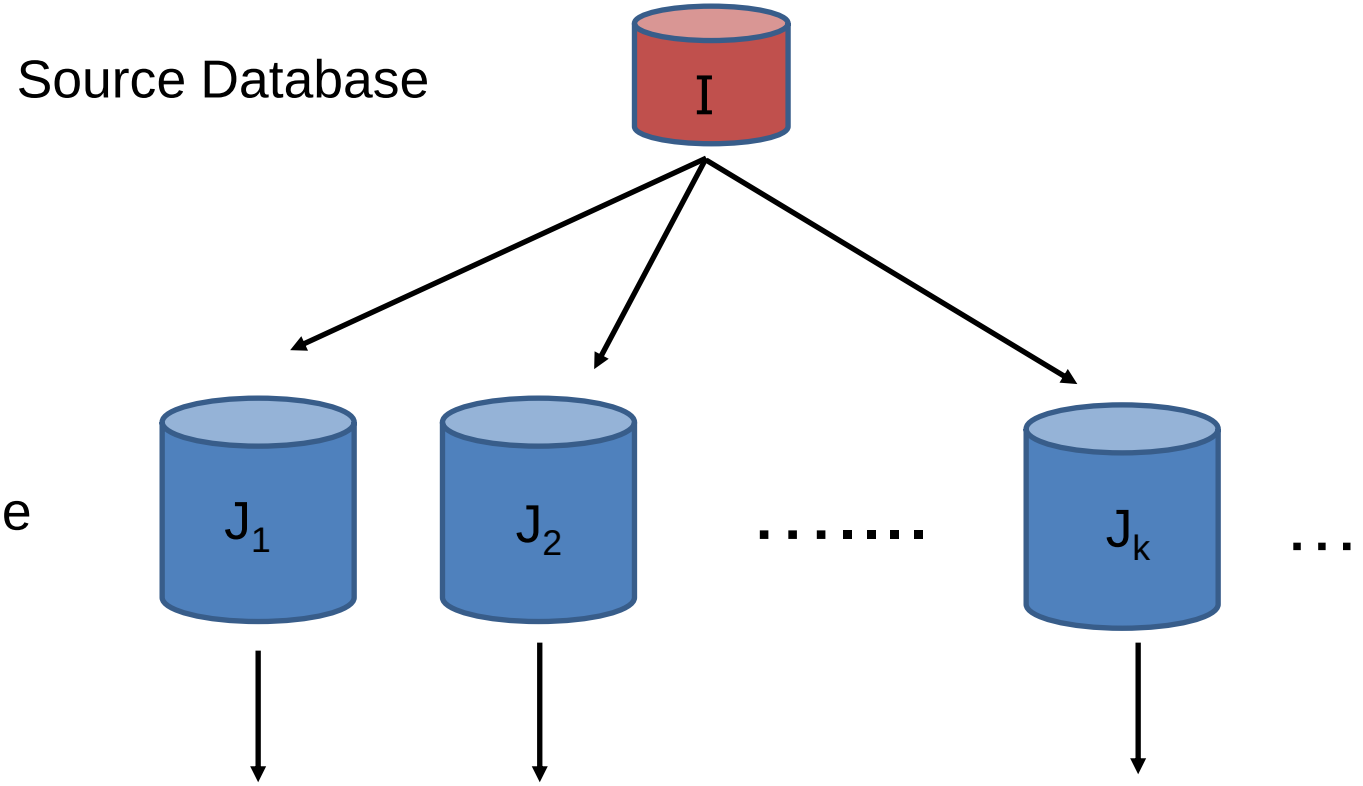
Definition: The certain answers of a query q over **T** on I

CERTAIN(q, **Sol** (I)) = ∩ { q(J): J is a solution for I }.

Note:

- Here, the possible worlds are the solutions for I.
- The number of solutions may be infinite (unlike subset repairs)

Visualizing the Certain Answers



Sol(I) = Set of the solutions of I

Query q

CERTAIN(q, **Sol**(I)) = $q(J_1) \cap q(J_2) \cap \ldots\ldots \cap q(J_k) \cap \ldots$

Schema-Mapping Specification Languages

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

- Question: How are schema mappings specified?

- Answer: Use a high-level, declarative language. In particular, it is natural to try to use relational calculus (first-order logic) as specification language for schema mappings.

- Fact: There is a fixed relational calculus sentence specifying a schema mapping **M*** such that **Sol(M*)** is undecidable.
  - Reason: Undecidability of the Finite Validity Problem.

Schema-Mapping Specification Languages: Bottom-Up

Let us consider some simple tasks that we should be able to support:

- Copy (Nicknaming):
  Copy each source table to a target table and rename it.
- Projection:
  Form a target table by projecting on some columns of a source table.
- Decomposition:
  Decompose a source table into two or more target tables.
- Column Augmentation:
  Form a target table by adding columns to a source table.
- Join:
  Form a target table by joining two or more source tables.
- Combinations of the above (e.g., "join + column augmentation")

Schema Mapping Specification Languages

- Copy (Nicknaming):
  - $\forall x_1, \ldots, x_n (P(x_1,\ldots,x_n) \rightarrow R(x_1,\ldots,x_n))$
- Projection:
  - $\forall x,y,z (P(x,y,z) \rightarrow R(x,y))$
- Decomposition:
  - $\forall x,y,z\ (P(x,y,z) \rightarrow R(x,y) \wedge T(y,z))$
- Column Augmentation:
  - $\forall x,y\ (P(x,y) \rightarrow \exists\ z\ R(x,y,z))$
- Join:
  - $\forall x,y,z (E(x,z) \wedge F(z,y) \rightarrow R(x,y,z))$
- Combinations of the above (e.g., "join + column augmentation")
  - $\forall x,y,z (E(x,z) \wedge F(z,y) \rightarrow \exists\ w\ T(x,y,z,w)))$

Schema Mapping Specification Languages

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

- Answer:
  - They can be specified using
    tuple-generating dependencies (tgds).

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

Schema Mapping Specification Language

The relationship between source and target is given by formulas of relational calculus, called

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

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

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

They are also known as GLAV (Global-and-Local-as-View) constraints

Example:

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

(here, we have dropped the universal quantifiers in front of s-t tgds)

Target Dependencies

In addition to source-to-target dependencies, we also consider target dependencies, since, after all, the target schema may have its own integrity constraints:

- 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)

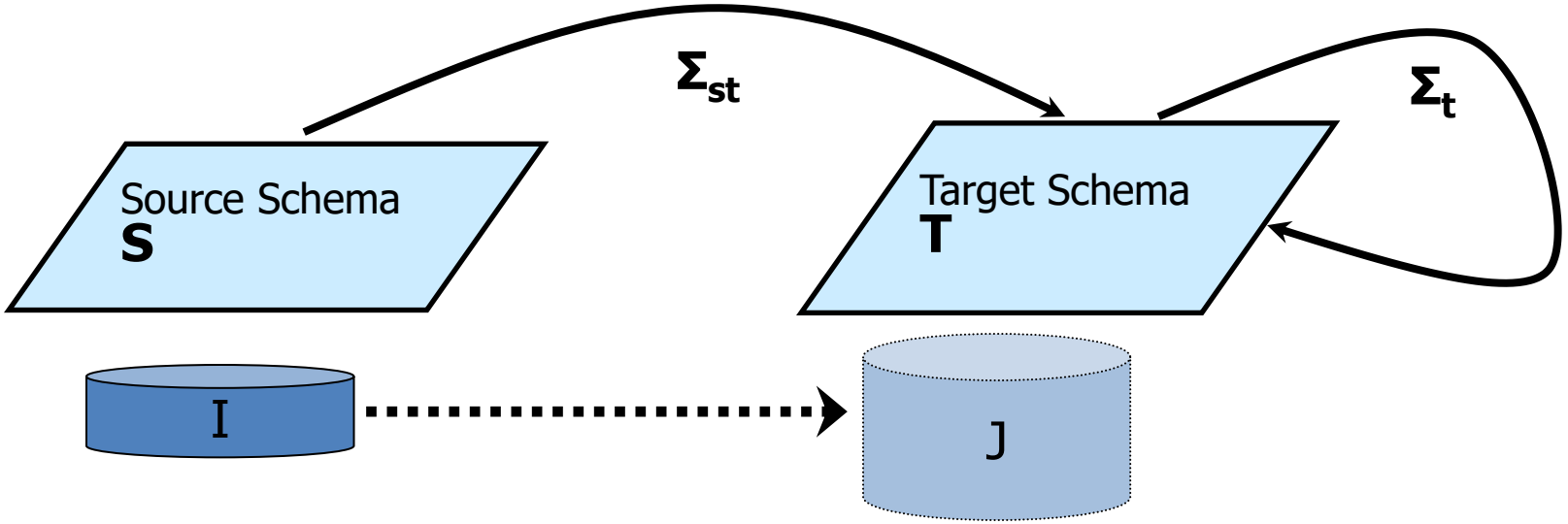
- Target Equality Generating Dependencies (egds):

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

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

  (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

Schema Mappings: An Example

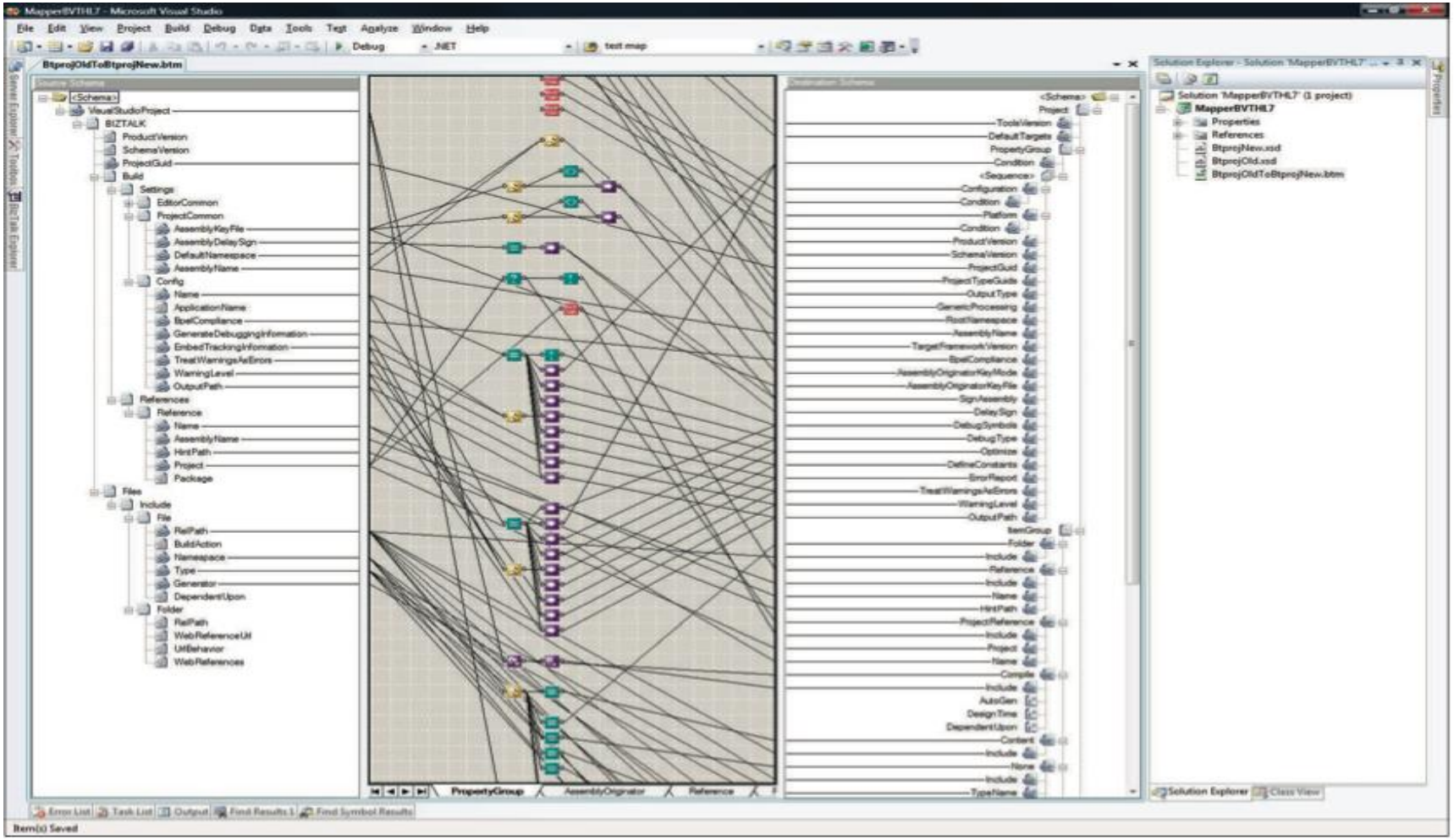
- Source Schema **S**: Movies database with relations
  P(title, year), R(title, director)
- Target Schema **T**: Movies database with relations
  Movies(title, year, director), Reviews(title, year, critic, score)

- Σ_{st} consists of the following source-to-target tgds
  - $\forall t\, \forall y\, \forall d\, (P(t,y) \wedge R(t,d) \rightarrow Movies(t,y,d))$
  - $\forall t\, \forall y\, (P(t,y) \rightarrow \exists c\, \exists s\, Reviews(t,y,c,s))$

- Σ_t consists of the following target egds and target tgds
  - $\forall t\, \forall y\, \forall d\, \forall d'\, (Movies(t,y,d) \wedge Movies(t,y,d') \rightarrow d = d')$
  - $\forall t\, \forall y\, \forall c\, \forall s\, \forall s'\, (Reviews(t,y,c,s) \wedge Reviews(t,y,c,s') \rightarrow s = s')$
  - $\forall t\, \forall y\, \forall c\, \forall s\, (Reviews(t,y,c,s) \rightarrow \exists d\, Movies(t,y,d))$

Visual Specification

- Screenshot from Bernstein and Haas 2008 CACM article. "*Information Integration in the Enterprise*"



Schema Mappings (one of many pages)

```
Map 2:
        for  sm2x0 in S0.dummy_COUNTRY_4
        exists  tm2x0 in S27.dummy_country_10, tm2x1 in S27.dummy_organiza_13
           where  tm2x0.country.membership=tm2x1.organization.id,
        satisf  sm2x0.COUNTRY.AREA=tm2x0.country.area, sm2x0.COUNTRY.CAPITAL=tm2x0.country.capital,
                sm2x0.COUNTRY.CODE=tm2x0.country.id, sm2x0.COUNTRY.NAME=tm2x0.country.name,
                sm2x0.COUNTRY.POPULATION=tm2x0.country.population,(
        Map 3:
                for  sm3x0 in S0.dummy_GEO_RIVE_23, sm3x1 in S0.dummy_RIVER_24,
                        sm3x2 in S0.dummy_PROVINCE_5
                   where  sm3x0.GEO_RIVER.RIVER=sm3x1.RIVER.NAME, sm3x2.PROVINCE.NAME=sm3x0.GEO_RIVER.PROVINCE,
                        sm3x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
                exists  tm3x0 in S27.dummy_river_24, tm3x1 in tm3x0.river.dummy_located_23,
                        tm3x4 in S27.dummy_country_10, tm3x5 in tm3x4.country.dummy_province_9,
                        tm3x6 in S27.dummy_organiza_13
                where  tm3x4.country.membership=tm3x6.organization.id, tm3x5.province.id=tm3x1.located.province,
                        tm2x0.country.id=tm3x1.located.country,
                satisf  sm2x0.COUNTRY.AREA=tm3x4.country.area, sm2x0.COUNTRY.CAPITAL=tm3x4.country.capital,
                        sm2x0.COUNTRY.CODE=tm3x4.country.id, sm2x0.COUNTRY.NAME=tm3x4.country.name,
                        sm2x0.COUNTRY.POPULATION=tm3x4.country.population, sm3x1.RIVER.LENGTH=tm3x0.river.length,
                        sm3x0.GEO_RIVER.COUNTRY=tm3x1.located.country, sm3x0.GEO_RIVER.PROVINCE=tm3x1.located.province,
                        sm3x1.RIVER.NAME=tm3x0.river.name ),(
        Map 4:
                for  sm4x0 in S0.dummy_GEO_ISLA_25, sm4x1 in S0.dummy_ISLAND_26,
                        sm4x2 in S0.dummy_PROVINCE_5
                   where  sm4x0.GEO_ISLAND.ISLAND=sm4x1.ISLAND.NAME, sm4x2.PROVINCE.NAME=sm4x0.GEO_ISLAND.PROVINCE,
                        sm4x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
                exists  tm4x0 in S27.dummy_island_26, tm4x1 in tm4x0.island.dummy_located_25,
                        tm4x4 in S27.dummy_country_10, tm4x5 in tm4x4.country.dummy_province_9,
                        tm4x6 in S27.dummy_organiza_13
                   where  tm4x4.country.membership=tm4x6.organization.id, tm4x5.province.id=tm4x1.located.province,
                        tm2x0.country.id=tm4x1.located.country,
                satisf  sm2x0.COUNTRY.AREA=tm4x4.country.area, sm2x0.COUNTRY.CAPITAL=tm4x4.country.capital,
                        sm2x0.COUNTRY.CODE=tm4x4.country.id, sm2x0.COUNTRY.NAME=tm4x4.country.name,
                        sm2x0.COUNTRY.POPULATION=tm4x4.country.population, sm4x1.ISLAND.AREA=tm4x0.island.area,
                        sm4x1.ISLAND.COORDINATESLAT=tm4x0.island.latitude, sm4x0.GEO_ISLAND.COUNTRY=tm4x1.located.country,
                        sm4x0.GEO_ISLAND.PROVINCE=tm4x1.located.province, sm4x1.ISLAND.COORDINATESLONG=tm4x0.island.longitude,
                        sm4x1.ISLAND.NAME=tm4x0.island.name ),(
        Map 5:
                for  sm5x0 in S0.dummy_GEO_SEA_19, sm5x1 in S0.dummy_SEA_20,
                        sm5x2 in S0.dummy_PROVINCE_5
                   where  sm5x2.PROVINCE.NAME=sm5x0.GEO_SEA.PROVINCE, sm5x0.GEO_SEA.SEA=sm5x1.SEA.NAME,
                                sm5x2.PROVINCE.COUNTRY=sm2x0.COUNTRY.CODE,
                exists  tm5x0 in S27.dummy_sea_19, tm5x1 in tm5x0.sea.dummy_located_18,
                        tm5x4 in S27.dummy_country_10, tm5x5 in tm5x4.country.dummy_province_9,
                        tm5x6 in S27.dummy_organiza_13
                   where  tm5x4.country.membership=tm5x6.organization.id, tm5x5.province.id=tm5x1.located.province,
                        tm2x0.country.id=tm5x1.located.country,
                satisf  sm2x0.COUNTRY.AREA=tm5x4.country.area, sm2x0.COUNTRY.CAPITAL=tm5x4.country.capital,
                        sm2x0.COUNTRY.CODE=tm5x4.country.id, sm2x0.COUNTRY.NAME=tm5x4.country.name,
                        sm2x0.COUNTRY.POPULATION=tm5x4.country.population, sm5x1.SEA.DEPTH=tm5x0.sea.depth,
                        sm5x0.GEO_SEA.COUNTRY=tm5x1.located.country, sm5x0.GEO_SEA.PROVINCE=tm5x1.located.province,
                        sm5x1.SEA.NAME=tm5x0.sea.name ),(
```

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)}

    (labelled nulls):
  - 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: X, Y, …

Main issues in data exchange

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

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

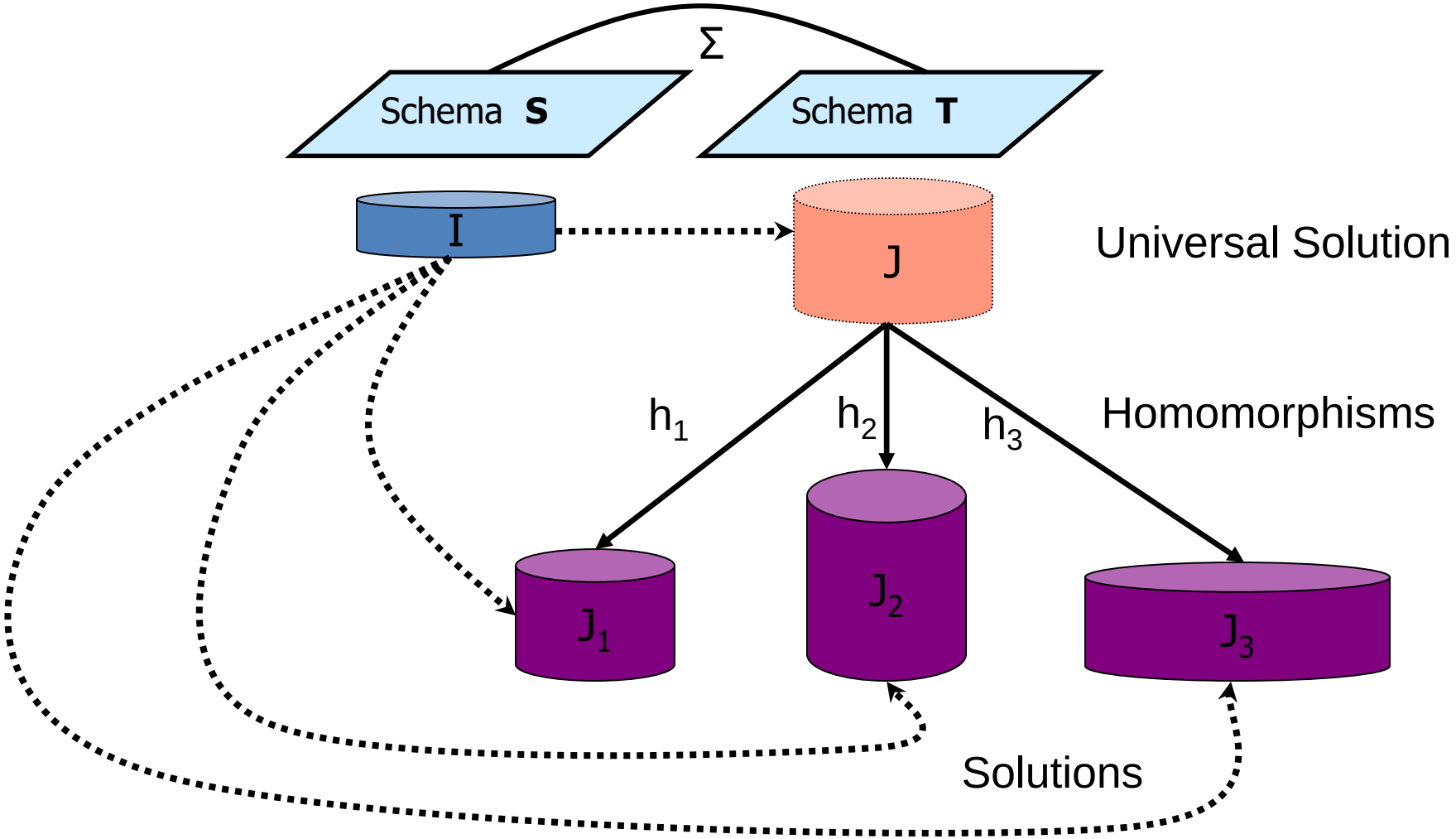
Universal Solutions in Data Exchange

Definition (Fagin, K ..., Miller, Popa 2005): 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.

Fact: Universal solutions have become the *preferred* solutions in data exchange.

Universal Solutions in Data Exchange



Example - continued

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

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

Source instance I = {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 tgds and egds?

Answer: Depending on the target constraints in Σ_t:

- **Sol(M)** can be trivial (solutions always exist).

  …
- **Sol(M)** can be in P.

  …
- **Sol(M)** can be undecidable.

Algorithmic Problems in Data Exchange

Proposition: Let **M** = (**S**, **T**, Σ_{st}) be a schema mapping with no target constraints, i.e., Σ_{st} is a set of s-t tgds and $\Sigma_t = \emptyset$. Then

- Solutions always exist; hence, **Sol(M)** is trivial.
- Universal solutions can be computed in polynomial time via the naïve chase procedure.

The Naïve Chase Algorithm

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}) (here $\Sigma_t = \emptyset$)

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

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

- If E= { (1,2) }, then F*={ (1,N),(N,2) } is universal solution for E
- If E ={ (1,2),(2,3),(1,4) }, then
  F*={ (1,M),(M,2),(2,N),(N,3),(1,U),(U,4) } is universal solution for E.

The Naïve Chase Algorithm

Example: Collapsing paths of length 2 to edges

M = (**S**, **T**, Σ_{st}) (here $\Sigma_t = \emptyset$)

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

- E = { (1,3), (2,4), (3,4) }
  F* = { (1,4) } Universal Solution for E

- E = { (1,3}, (2,4), (3,4), (4,3) }
  F* = { (1,4), (2,3), (3,3), (4,4) } Universal solution for E

Algorithmic Problems in Data Exchange

Question:

What about arbitrary target tgds and egds?

More formally:

What can we say about the existence-of-solutions problem for schema mappings **M**= (**S**, **T**, Σ^*_{st}, Σ^*_t) such that

- Σ^*_{st} is a set of s-t tgds;
- Σ^*_t is a set of target tgds and target egds?

Undecidability in Data Exchange

Theorem (K …, Panttaja, Tan - 2006):

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 and two target tgds;
- The existence-of-solutions problem **Sol(M)** is undecidable.

Hint of Proof: Reduction from the

Embedding Problem for Finite Semigroups:

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

Semigroups

Definition: A semigroup is a structure of the form **A** = (A, $\star$), where A is a set and $\star : A \times A \rightarrow A$ is a binary associative operation, i.e.,

$$a \star (b \star c) = (a \star b) \star c \text{ holds for all a, b, c in A.}$$

- If $\star$ is a partial function and $a \star (b \star c) = (a \star b) \star c$ holds whenever $\star$ is defined then **A** = (A, $\star$) is a partial semigroup.

Embedding Problem for Finite Semigroups:

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

- Informally,

Can a given partial multiplication table be completed to a total multiplication table satisfying associativity?

The Embedding Problem for Finite Semigroups

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.

Corollary: The Embedding Problem for Finite Semigroups is undecidable.

The Embedding Problem & Data Exchange

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

(universal quantifiers dropped from the constraints below)

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

Existence-of-Solutions for Schema Mappings

Summary: The existence-of-solutions problem

- is trivial for schema mappings with only source-to-target tgds (no target dependencies).

- is undecidable for schema mappings in which the target dependencies are arbitrary tgds and egds;

Question: Are there rich classes of target tgds and egds for which the existence-of-solutions problem is decidable (in fact, tractable)?

Algorithmic Properties of Universal Solutions

Theorem (Fagin, K ..., Miller, Popa - 2005):

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

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

Then:

- Universal solutions exist if and only if solutions exist.

- **Sol(M)** is in P.

- A *canonical* universal solution (if a solution exists) 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 (no existential quantifiers in the right-hand side)

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

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

- Acyclic sets of inclusion dependencies

  Large class of dependencies occurring in practice.

Weakly Acyclic Sets of Tgds: Definition

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

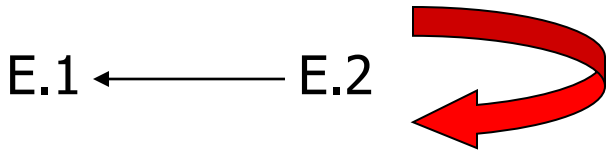
- Σ is weakly acyclic if the position graph has no cycle containing a special edge.

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

Weakly Acyclic Sets of Tgds: Examples

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

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



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

   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

$E(1,2) \Rightarrow$

$E(2,X_1) \Rightarrow$

$E(X_1,X_2) \Rightarrow$

$E(X_2, X_3) \Rightarrow$

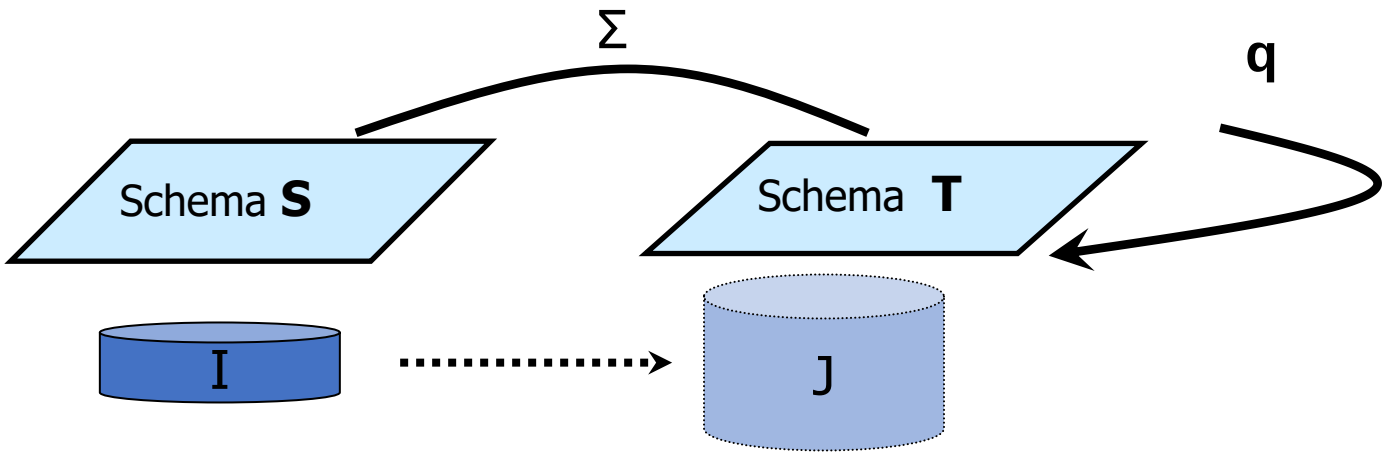
…

infinite chase

The Complexity of the Existence of Solutions

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

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

CERTAIN(q, **Sol**(I)) = ∩ { q(J): J is a solution for I }.

Note: Here, the possible worlds are the solutions for I.

Certain Answers in Data Exchange

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

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

Target conjunctive query $q(x) :- \exists y\, H(x,y)$

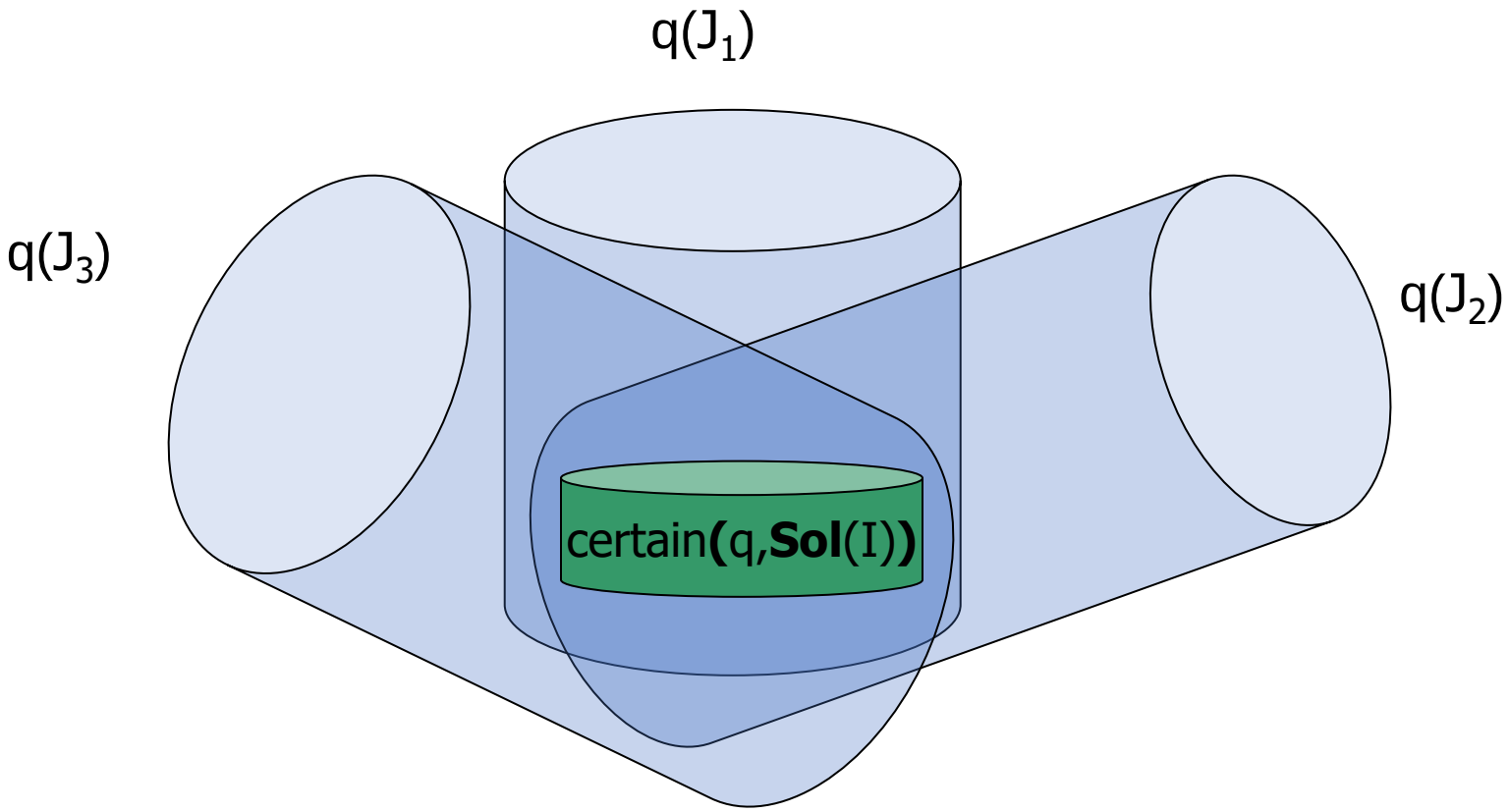
Source instance I = {E(a,b)}

Solutions: Infinitely many solutions exist

- J_1 = {H(a,b), H(b,b)} $q(J_1)$ = {a, b}
- J_2 = {H(a,a), H(a,b)} $q(J_2)$ = {a}
- J_3 = {H(a,X), H(X,b)} $q(J_3\}$ = {a, X}
- J_4 = {H(a,X), H(X,b), H(a,Y), H(Y,b)} $q(J_4)$ = {a, X, Y}
- J_5 = {H(a,X), H(X,b), H(Z,Z)} $q(J_5)$ = {a,X,Z}
- …

• CERTAIN(q, **Sol**(I)) = $\bigcap$ { q(J): J is a solution for I } = {a}

Certain Answers Semantics



CERTAIN(q, **Sol**(I)) = ⋂ { 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, **Sol**(I)) = the set of all "variable-free" tuples in q(J).

- CERTAIN(q, **Sol**(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 "variables".

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

Computing the Certain Answers

Theorem (FKMP): Schema mapping **M** = (**S**, **T**, Σ_{st}, Σ_t), query q that is a union of conjunctive queries over **T**.
If I is a source instance and J is a universal solution for I, then
CERTAIN(q,**Sol**(I)) = the set of all "variable-free" tuples in q(J).
Proof:
Step1: Show that CERTAIN(q, **Sol**(I)) consists of tuples having only constants (no variables) - Exercise.

Step 2: Since J is a solution, we have that CERTAIN(q,**Sol**(I)) $\subseteq$ q(J).

Step 3: Let $(a_1,..,a_k)$ be a tuple of constants in q(J). Let J' be an arbitrary solution for I w.r.t. **M**. Then there is a homomorphism h: J $\rightarrow$ J' that is the identity on constants.
Since q is a union of conjunctive queries, h is a homomorphism, and $(a_1,..,a_k) \in q(J)$, we have that $(h(a_1),..,h(a_k)) \in q(J')$.
Since $h(a_1) = a_1$, …, $h(a_k) = a_k$, we have that $(a_1,\ldots,a_k) \in q(J')$.
Since J' was an arbitrary solution, we have that $(a_1,\ldots,a_k) \in$ CERTAIN(q,I) ,
Hence q(J) $\subseteq$ CERTAIN(q,I) .

Certain Answers in Data Exchange

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

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

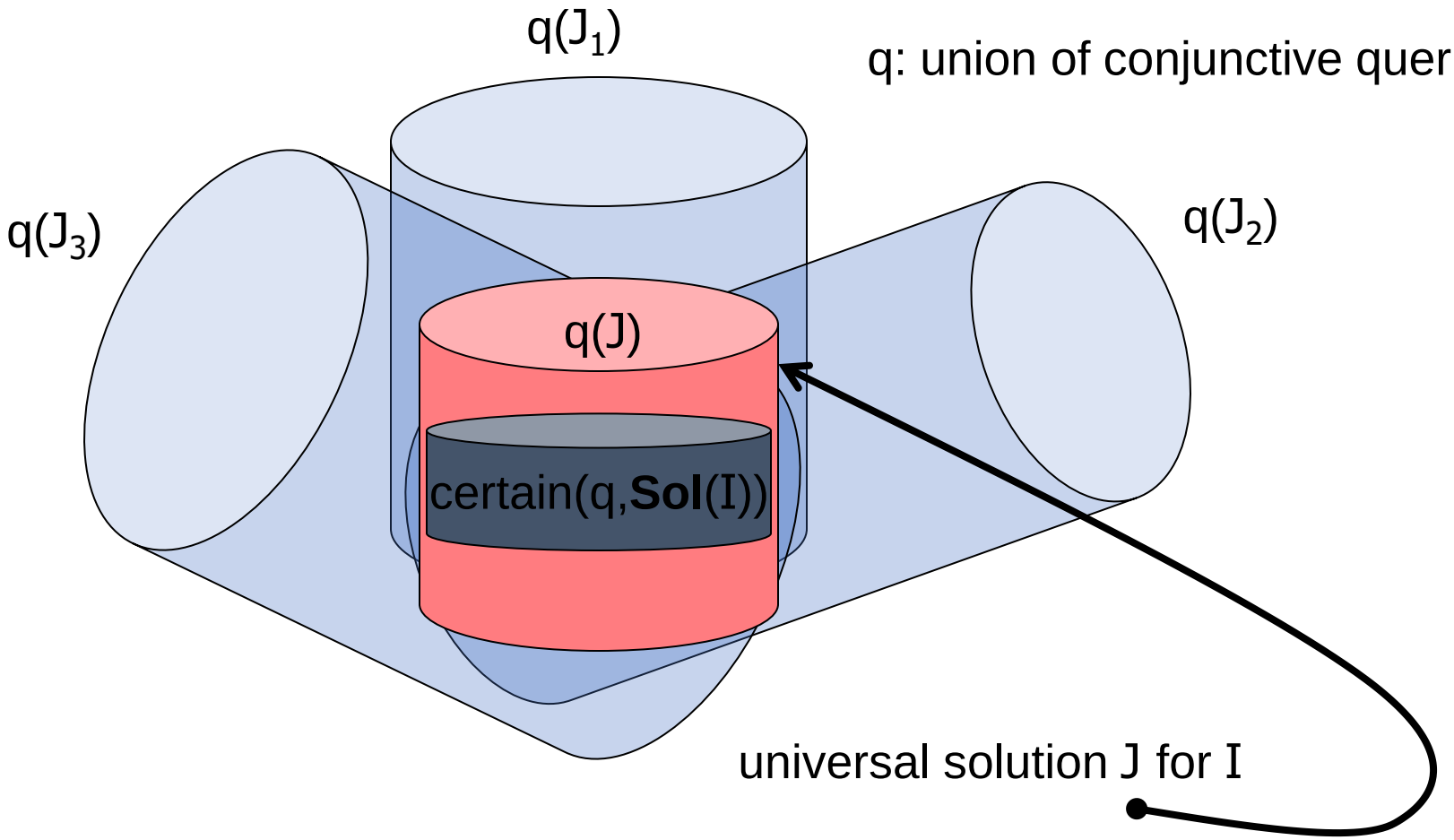
Target conjunctive query $q(x)$:- $H(x,y)$

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

Solutions: Infinitely many solutions exist

- $J_1 = \{H(a,b), H(b,b)\}$ $\quad q(J_1) = \{a, b\}$
- $J_2 = \{H(a,a), H(a,b)\}$ $\quad q(J_2) = \{a\}$
- $J_3 = \{H(a,X), H(X,b)\}$ universal solution
  - $q(J_3\} = \{a, X\}$
  - Variable-free part of $q(J_3) = \{a\}$ = CERTAIN(q, **Sol**(I))

Certain Answers via Universal Solutions



CERTAIN(q,**Sol**(I)) = set of null-free tuples of q(J).

Data Complexity of Conjunctive Queries

Setting	Data Complexity
Single DB	LOGSPACE (also in P)
CERTAIN(q,I) : Inconsistent Databases (FDs, subset repairs)	coNP-complete
certain(q,I): Data Exchange s-t tgds, weakly acyclic t-tgds, t-egds	PTIME
certain(q,I): Data Exchange s-t tgds, arbitrary t-tgds, t-egds	Undecidable, in general