Data Exchange

V. Efthymiou
vefthym@csd.uoc.gr
http://www.csd.uoc.gr/~hy562
University of Crete, Fall 2020

slides kindly offered by Prof. Phokion Kolaitis from his UCSC course CSE277
The Data Interoperability Problem

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

bullet Applications need to access all these data!

bullet Two different, but closely related, facets of data interoperability:
  - Data Exchange (aka Data Translation)
  - Data Integration (aka Data Federation, Entity Resolution, …)
What is 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

“Data Exchange is the oldest database problem” Phil Bernstein – 2003

Data Exchange underlies:

- Data Warehousing, ETL (Extract-Transform-Load) tasks
- XML publishing, XML storage, …
Challenges in Data Interoperability

- Data interoperability tasks require expertise, effort, and time
- Human experts have to generate complex transformations that specify the relationships between schemas written as programs or as scripts
- At present, there is relatively little automation

**Question**: How can we address these challenges?

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

High-level, declarative assertions that specify the relationships 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:
  - easier to generate and manage (semi-)automatically
  - can be compiled into SQL/XSLT scripts automatically
**Schema Mappings**

- **Schema Mapping** $M = (S, T, \Sigma)$
  - Source schema $S$, Target schema $T$
  - A set $\Sigma$ of high-level, declarative assertions (constraints) that specify the relationships between $S$-instances and $T$-instances
Schema Mappings & Data Exchange

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

- **Data Exchange** via the schema mapping $M = (S, T, \Sigma)$
  - Transform a given source instance $I$ to a target instance $J$, so that $(I, J)$ satisfy the specifications $\Sigma$ of $M$
Solutions in Schema Mappings

Definition: Schema Mapping \( M = (S, T, \Sigma) \)
If \( I \) is a source instance, then a solution for \( I \) is a target instance \( J \) such that \( (I, J) \) satisfy \( \Sigma \)

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: Basic Problems

Definition: Schema Mapping \( M = (S, T, \Sigma) \)

- The existence-of-solutions problem \( \text{Sol}(M) \): (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
Simple tasks that a schema mapping specification language should support:

- **Copy (Nicknaming):**
  - Copy each source table to a target table and rename it

- **Projection:**
  - Form a target table by projecting on one or more columns of a source table

- **Decomposition:**
  - Decompose a source table into two or more target tables

- **Column Augmentation:**
  - Form a target table by adding one or more 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

Simple tasks that a schema mapping specification language should support:

- **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:**
  $$\forall x, y, z (P(x, y, z) \rightarrow R(x, y) \land T(y, z))$$

- **Column Augmentation:**
  $$\forall x, y (P(x, y) \rightarrow \exists z R(x, y, z))$$

- **Join:**
  $$\forall x, y, z (E(x, z) \land F(z, y) \rightarrow R(x, y, z))$$

- **Combinations of the above (e.g., “join + column augmentation”)**
  $$\forall x, y, z (E(x, z) \land F(z, y) \rightarrow \exists w T(x, y, z, w))$$
### Schema Mapping
### Specification Languages

<table>
<thead>
<tr>
<th>Drug</th>
<th>Medication</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td>did</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>act. ingred.</td>
<td>act. ingred.</td>
</tr>
<tr>
<td>1</td>
<td>Depon</td>
</tr>
<tr>
<td></td>
<td>Paracetamol</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td></td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Copy (Nicknaming):

\[ \forall d, n, a (\text{Drug}(d, n, a) \rightarrow \text{Medication}(d, n, a)) \]
Schema Mapping
Specification Languages

<table>
<thead>
<tr>
<th>Drug</th>
<th>did</th>
<th>name</th>
<th>act.ingred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td>name</td>
<td>act.ingred.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Depon</td>
<td>Paracetamol</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
<td>Ibuprofen</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Medication</th>
<th>did</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Depon</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Projection:
\[ \forall d, n, a(Drug(d, n, a) \rightarrow Medication(d, n)) \]
Schema Mapping
Specification Languages

**Medication**
<table>
<thead>
<tr>
<th>did</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depon</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Ingredients**
<table>
<thead>
<tr>
<th>did</th>
<th>act. ingred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Paracetamol</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Decomposition:**

$$\forall d, n, a (\text{Drug}(d, n, a) \rightarrow \text{Medication}(d, n) \land \text{Ingredients}(d, a))$$
Schema Mapping
Specification Languages

Drug

<table>
<thead>
<tr>
<th>did</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depon</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Medication

<table>
<thead>
<tr>
<th>did</th>
<th>name</th>
<th>act_ingred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depon</td>
<td>value1</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
<td>value2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Column Augmentation:

\[ \forall d, n (\text{Drug}(d, n) \rightarrow \exists a \text{ Medication}(d, n, a)) \]
### Medication

<table>
<thead>
<tr>
<th>did</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depon</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Ingredients

<table>
<thead>
<tr>
<th>did</th>
<th>act. ingred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Paracetamol</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Drug

<table>
<thead>
<tr>
<th>did</th>
<th>name</th>
<th>act. ingred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depon</td>
<td>Paracetamol</td>
</tr>
<tr>
<td>2</td>
<td>Ibuprofen</td>
<td>Ibuprofen</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Join:**

\[ \forall d, n, a (\text{Medication}(d, n) \land \text{Ingredients}(d, a) \rightarrow \text{Drug}(d, n, a)) \]
Question: What do all these tasks (copy, projection,…) have in common?

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

● In fact, they can be specified using a special class of tgds known as source-to-target tuple generating dependencies (st-tgds)
Database Dependencies

- Database dependencies are semantic restrictions on databases
- Codd introduced functional dependencies in 1972
- Soon after this, several different classes of integrity constraints were introduced:
  - Multi-valued Dependencies
  - Join Dependencies
  - Inclusion Dependencies
  - …
- 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
Definition: A tuple-generating dependency (tgd) is a formula of relational calculus of the form:

\[ \forall x_1, \ldots, x_n \left( \varphi(x_1, \ldots, x_n) \rightarrow \exists y_1, \ldots, y_m \left( \psi(x'_1, \ldots, x'_k, y_1, \ldots, y_m) \right) \right), \]

where

- \( \varphi(x_1, \ldots, x_n) \) and \( \psi(x'_1, \ldots, x'_k, y_1, \ldots, y_m) \) are conjunctions of atomic formulas
- The variables \( x'_1, \ldots, x'_k \) are among the variables \( x_1, \ldots, x_n \)
Equality-Generating Dependencies

Definition: A equality-generating dependency (egd) is a formula of relational calculus of the form:

$$\forall x_1, \ldots, x_n (\varphi(x_1, \ldots, x_n) \rightarrow x_1 = x_j),$$

where $\varphi(x_1, \ldots, x_n)$ is a conjunction of atomic formulas

Example:

$$\forall d, x, y (Ingredients(d, x) \land Ingredients(d, y) \rightarrow x = y)$$

(which implies that you cannot have two different active ingredients for the same drug)
Schema Mapping
Specification Languages

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

Source-to-Target Tuple Generating Dependencies (st-tgds)
\[ \forall x (\varphi(x) \rightarrow \exists y \psi(x, y)) \], where
- \( \varphi(x) \) is a conjunction of atoms over the source
- \( \psi(x, y) \) is a conjunction of atoms over the target
- \( x \) and \( y \) are tuples of variables
  - i.e, \( x = (x_1, \ldots, x_n) \), \( y = (y_1, \ldots, y_m) \)

Example:
\( \text{Student}(s) \land \text{Enrolls}(s,c) \rightarrow \exists t \exists g (\text{Teaches}(t,c) \land \text{Grade}(s,c,g)) \)
(here we have dropped the universal quantifiers in front of st-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:** \( \varphi_T(x) \rightarrow \exists y \psi_T(x, y) \)

Example: Dept \((\text{did, dname, mgr_id, mgr_name}) \rightarrow \text{Mgr (mgr_id, did)}\)  
(a target inclusion dependency constraint)

- **Target Egds:** \( \varphi_T(x) \rightarrow (x_1 = x_2) \)

Example: \((\text{Mgr (e, } d_1) \land \text{Mgr (e, } d_2) \rightarrow d_1 = d_2)\)  
(a target key constraint)
Data Exchange Framework

Schema Mapping $M = (S, T, \Sigma_{st}, \Sigma_t)$, where

- $\Sigma_{st}$ is a set of source-to-target tgds
- $\Sigma_t$ is a set of target tgds and target egds
Underspecification in Data Exchange

Given a source instance, multiple solutions may exist

Example:
Source relation \( E(A,B) \), target relation \( H(A,B) \)
\[
\Sigma: \quad E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y))
\]
Source instance \( I = \{E(a,b)\} \)

Solutions: Infinitely many solutions exist

- \( J_1 = \{H(a,b), H(b,b)\} \)
- \( J_2 = \{H(a,a), H(a,b)\} \) constants: \( a, b, \ldots \)
- \( J_3 = \{H(a,X), h(X,b)\} \) variables (labelled nulls): \( X, Y, \ldots \)
- \( J_4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\} \)
- \( J_5 = \{H(a,X), H(X,b), H(Y,Y)\} \)
Main Issues in Data Exchange

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

- When more than one solutions exist, which solutions are “better” than others?

- How do we compute a “best” solution?

- In other words, what is the “right” semantics of data exchange?
Universal Solutions in Data Exchange

**Definition** (FKMP 2003): 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** (labelled nulls): other entries in target instances
- **Homomorphism** $h: J_1 \rightarrow J_2$ between target instances:
  - $h(c) = c$, for constant $c$
  - If $P(a_1, \ldots, a_m)$ is in $J_1$, then $P(h(a_1), \ldots, h(a_m))$ is in $J_2$

**Claim**: Universal solutions are the preferred solutions in data exchange
Universal Solutions in Data Exchange

Diagram: Two schemas, S and T, are connected through a universal solution, J. Homomorphisms h₁, h₂, and h₃ map I to J₁, J₂, and J₃, which are solutions.
Example - continued

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

\[ \Sigma: \quad \text{E}(x,y) \rightarrow \exists z \ (\text{H}(x,z) \land \text{H}(z,y)) \]

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

**Solutions:** *Infinitely* many solutions exist

- **J_1** = \{\text{H}(a,b), \text{H}(b,b)\}  
  is not universal

- **J_2** = \{\text{H}(a,a), \text{H}(a,b)\}  
  is not universal

- **J_3** = \{\text{H}(a,X), \text{H}(X,b)\}  
  is universal

- **J_4** = \{\text{H}(a,X), \text{H}(X,b), \text{H}(a,Y), \text{H}(Y,b)\} 
  is universal

- **J_5** = \{\text{H}(a,X), \text{H}(X,b), \text{H}(Y,Y)\}  
  is not universal

**Reminder:** 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 \( J_2 \)
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

- **Representations 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 \( \text{Sol}(M) \) for a fixed schema mapping \( M = (S, T, \Sigma_{st}, \Sigma_t) \) specified by st-tgds, and target tgds and egds?

**Answer:** Depending on the target constraints in \( \Sigma_t \):
- \( \text{Sol}(M) \) can be trivial (solutions always exist)
- ...
- \( \text{Sol}(M) \) can be in PTIME
- ...
- \( \text{Sol}(M) \) can be undecidable
Algorithmic Problems in Data Exchange

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

- Solutions always exist; hence, $\text{Sol}(M)$ is trivial
- Universal solutions can be computed in polynomial time via the naïve chase procedure
THE CHASE ALGORITHM
The Naïve Chase Algorithm

Naïve chase Algorithm for $M^* = (S, T, \Sigma_{st})$: given a source instance $I$, build a target instance $J^*$ that satisfies each st-tgd in $\Sigma_{st}$

- by introducing new facts in $J^*$ as dictated by the RHS of the st-tgd and
- by introducing new values (variables) in $J^*$ each time existential quantifiers need witnesses

Example: $M = (S, T, \Sigma_{st})$ (here $\Sigma_t = \emptyset$)

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

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

\[ M = (S, T, \Sigma_{st}) \quad \text{ (here } \Sigma_t = \emptyset) \]

\[ \Sigma_{st}: \forall (x,z) \land \exists (z,y) \rightarrow F(x,y) \]

\begin{itemize}
  \item \( E = \{ (1,3), (2,4), (3,4) \} \)
  \[ F = \{ (1,4) \} \quad \text{Universal solution for } E \]

  \item \( E = \{ (1,3), (2,4), (3,4), (4,3) \} \)
  \[ F = \{ (1,4), (2,3), (3,3), (4,4) \} \quad \text{Universal solution for } E \]
\end{itemize}
Question:
What can we say about the existence-of-solution problem from schema mappings $M = (S, T, \Sigma^*_st, \Sigma^*_t)$ such that

- $\Sigma^*_st$ is a set of st-tgds;
- $\Sigma^*_t$ is a set of target tgds and target egds?
The Existence-of-solution Problem 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 and, in fact, tractable?
Algorithmic Properties of Universal Solutions

Theorem (FKMP 2003): Schema mapping $M = (S, T, \Sigma^*_{st}, \Sigma^*_t)$ such that:

- $\Sigma_{st}$ is a set of st-tgds;
- $\Sigma_t$ 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
- $\text{Sol}(M)$ is in PTIME
- 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** (i.e., tgds with no existential quantifiers on the RHS)
  
  \[ \varphi_T(x, x') \rightarrow \psi_T(x), \]

  where \( \varphi_T(x, x') \) and \( \psi_T(x) \) 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 \( \Sigma \) of tgds:
  - **Nodes**: \( R.A \), with \( R \) relation symbol, \( A \) attribute of \( R \)
  - **Edges**: for every \( \varphi(x) \rightarrow \exists y \psi(x, y) \) in \( \Sigma \), for every \( x \) occurring in \( \psi \), for every occurrence of \( x \) in \( \varphi \) in \( R.A \):
    - For every occurrence of \( x \) in \( \psi \) in \( S.B \), add an edge \( R.A \rightarrow S.B \)
    - In addition, for every existentially quantified \( y \) that occurs in \( \psi \) in \( T.C \), add a special edge \( R.A \rightarrow T.C \)

- \( \Sigma \) is weakly acyclic if the position graph has no cycle containing a special edge

- A tgd \( \theta \) is weakly acyclic if so is the singleton set \( \{ \theta \} \)
Weakly Acyclic Sets of Tgds: Examples

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

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

**Edges**: for every \( \varphi(x) \rightarrow \exists y \psi(x,y) \) in \( \Sigma \), for every \( x \) occurring in \( \psi \), for every occurrence of \( x \) in \( \varphi \) in R.A:
- For every occurrence of \( x \) in \( \psi \) in S.B, add an edge R.A \rightarrow S.B
- For every existentially quantified \( y \) that occurs in \( \psi \) in T.C, add a special edge R.A \rightarrow T.C
Data Exchange with Weakly Acyclic Tgds

**Theorem** (FKMP): Schema mapping $M = (S, T, \Sigma_{st}, \Sigma_t)$ such that:

- $\Sigma_{st}$ is a set of st-tgds;
- $\Sigma_t$ 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 PTIME.
Chase Procedure for Tgds and Egds

Given a source instance I,

1. Use the naïve chase to chase I with $\Sigma_{st}$ and obtain a target instance $J^*$
2. Chase $J^*$ with the target tgds and the target egds in $\Sigma_t$ 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 st-tgd and introduce new values (variables) in $J$ each time existential quantifiers need witnesses
   
   2.2. For target egds $\varphi(x) \rightarrow x_1 = x_2$
      
      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

$E(1,2) \Rightarrow$
$E(2,X1) \Rightarrow$
$E(X1,X2) \Rightarrow$
$E(X2,X3) \Rightarrow$

... infinite chase
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 PTIME for schema mappings in which the set of the target dependencies is the union of a weakly acyclic set of tgds and a set of egds

**Note:**

- These are data complexity results
- The combined complexity of the existence-of-solutions problem is 2EXPTIME-complete
  (weakly acyclic sets of target tgds and egds)
# The Complexity of the Existence-of-solutions Problem

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Σ_{st} a set of st-tgds</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Σ_t = ∅</td>
<td>Trivial</td>
<td>Trivial</td>
<td>PTIME</td>
</tr>
<tr>
<td>No target constraints</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Σ_t: Weakly acyclic set of target tgds + egds</td>
<td>PTIME complete</td>
<td>PTIME Universal solutions exist if and only if solutions exist</td>
<td>PTIME</td>
</tr>
<tr>
<td>Σ_t: target tgds + egds</td>
<td>Undecidable in general</td>
<td>Undecidable in general</td>
<td>No algorithm exists, in general</td>
</tr>
</tbody>
</table>
Universal solutions need not be unique

**Question**: is there a “best” universal solution?

**Answer**: R. Fagin, P. Kolaitis and L. Popa 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 sub-instance $J'$ that is homomorphically equivalent to $J$

**Facts**:
- Every finite relational structure has a core
- The core is unique up to isomorphism
**The Core of a Structure**

**Definition:** $J'$ is the core of $J$ if

- $J' \subseteq J$
- there is homomorphism $h: J \rightarrow J'$
- there is no homomorphism $g: J \rightarrow J''$, where $J'' \subset J'$
Example - continued

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

$\Sigma$: \( E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y)) \)

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

Solutions: Infinitely many solutions exist

- \( J_3 = \{H(a,X), h(X,b)\} \) is the core

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

- \( J_5 = \{H(a,X), H(X,b), H(Y,Y)\} \) is not universal
Core: The Smallest Universal Solution

**Theorem** (FKP 2003): $M = (S, T, \Sigma_{st}, \Sigma_t)$ a schema mapping:
- All universal solutions have the same core
- The core of the universal solutions is the smallest universal solution
- If every target constraint is an egd, then the core is polynomial-time computable

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

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

**Note**: It is the standard semantics in data integration.
Certain Answers in Data Exchange

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

$\Sigma$: $E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y))$

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

**Solutions**: Infinitely many solutions exist

- $J_1 = \{H(a,b), H(b,b)\}$ \hspace{1cm} $q(J_1) = \{a,b\}$
- $J_2 = \{H(a,a), H(a,b)\}$ \hspace{1cm} $q(J_2) = \{a\}$
- $J_3 = \{H(a,X), h(X,b)\}$ \hspace{1cm} $q(J_3) = \{a,X\}$
- $J_4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\}$ \hspace{1cm} $q(J_4) = \{a,X,Y\}$
- $J_5 = \{H(a,X), H(X,b), H(Y,Y)\}$ \hspace{1cm} $q(J_5) = \{a,X,Z\}$

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

\[ \text{certain}(q,I) = \bigcap \{ q(J) : J \text{ is a solution for } I \} \]
Computing the Certain Answers

**Theorem (FKMP):** Schema mapping \( M = (S, T, \Sigma_{st}, \Sigma_t) \) such that:

- \( \Sigma_{st} \) is a set of st-tgds;
- \( \Sigma_t \) 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
  \[
  \text{certain}(q, I) = \text{the set of all "variable-free" tuples in } q(J)
  \]

- Hence, \( \text{certain}(q, I) \) is computable in time polynomial in \( |I| \):
  1. Compute a canonical universal solution \( J \) 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 \( M = (S, T, \Sigma_{st}, \Sigma_t) \) and a 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
\[
\text{certain}(q, I) = \text{the set of all “variable-free” tuples in } q(J)
\]

**Proof:**

**Step1:** Show that \( \text{certain}(q, I) \) consists of tuples having only constants (no variables) – Exercise

**Step2:** Since \( J \) is a solution, we have that \( \text{certain}(q, I) \subseteq q(J) \)

**Step3:** Let \((a_1, \ldots, a_k)\) 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 of constants. Since \( q \) is a union of conjunctive queries, \( h \) is a homomorphism, and \((a_1, \ldots, a_k) \in q(J)\), we have that \((h(a_1), \ldots, h(a_k)) \in q(J')\).

Since \( h(a_1) = a_1, \ldots, h(a_k) = a_k \), we have that \((a_1, \ldots, a_k) \in q(J')\). Since \( J' \) was an arbitrary solution, we have that \((a_1, \ldots, a_k) \in \text{certain}(q, I)\), hence \( q(J) \subseteq \text{certain}(q, I) \).
Certain Answers in Data Exchange

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

$\Sigma$: $E(x,y) \rightarrow \exists z \ (H(x,z) \land H(z,y))$

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 $q(J_2) = \{a\}$
- $J_3 = \{H(a,X), h(X,b)\}$ \quad universal solution
  - $q(J_3) = \{a,X\}$
  - Variable-free part of $q(J_3) = \{a\} = \text{certain}(q,I)$
Certain Answers via Universal Solutions

$q(J_1)$

$q: \text{union of conjunctive queries}$

$q(J_3)$

$q(J_2)$

$q(J)$

$\text{certain}(q,I)$

universal solution $J$ for $I$

$\text{certain}(q,I) = \text{set of null-free tuples of } q(J)$. 