Data exchange: transform data between different applications, each with its own schema

- An old problem (e.g., the EXPRESS system [SHT’77])
- Revisited and formalized by Fagin et al. [FKMP03] in 2003
  - Test-of-time award in ICDT 2013
- Lots of followups since 2013, including specialized sessions in main database conferences
- Implementation by the Clio project (IBM Almaden + University of Toronto), productized within IBM DB2
- Application to various data models (XML, RDF)
Introduction

Schema Mappings

Source/Target Instances

Answering Target Queries

Finding Universal Solutions

References

Background

Academic Impact

Popular Research Challenges

- How to materialize target data from source data?
- Answering target queries given source data
  - Semantics, complexity
  - Manipulating mapping specifications
  - Composition, inversion

The core of these challenges is missing information:
- underspecified logical rules, missing target attributes

Running Example

<table>
<thead>
<tr>
<th>Source: ManagedBy</th>
<th>emp</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td></td>
<td>Ahuva</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td></td>
</tr>
<tr>
<td>Ella</td>
<td></td>
<td>Doron</td>
</tr>
</tbody>
</table>

Target:

<table>
<thead>
<tr>
<th>Emp</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- No manager manages >1 departments
- Every manager is an employee

Table of Contents

- Introduction
- Schema Mappings
- Source/Target Instances
- Answering Target Queries
- Finding Universal Solutions

Example

**Definition (Schema Mapping) [FKMP05]**

A *schema mapping* is a triple \((S, T, \Sigma)\) where:

- \(S = \{S_1, \ldots, S_n\}\) is a *source schema*
- \(T = \{T_1, \ldots, T_m\}\) is a *target schema*
- \(S\) and \(T\) have no common relation names
- \(\Sigma\) is a set of logical formulas over \(S \cup T\)

\[ S: \text{ManagedBy} \]

\[ T: \text{Emp} \quad \text{Dept} \]

- If \(e\) is managed by \(m\), then \(e\) belongs to a department managed by \(m\)
- Every manager is an employee of her department
- No manager manages more than one department
We consider three types of constraints in \( \Sigma \):

- Source-to-target tuple-generating dependencies (st-TGDs)
- Target tuple-generating dependencies (t-TGDs)
- Target equality-generating dependencies (t-EGDs)

Let \((S, T, \Sigma)\) be a schema mapping.

Recall: a TGD is an expression of the form

\[
\forall x[\varphi(x) \rightarrow \exists y[y(x, y)]]
\]

where \(\varphi(x)\) and \(y(x, y)\) are conjunctions of atomic formulas \(R(t_1, \ldots, t_k)\).

In an st-TGD, \(\varphi(x)\) is over \(S\) (the source schema), and \(y(x, y)\) is over \(T\) (the target schema).

In a t-TGD, both \(\varphi(x)\) and \(y(x, y)\) are over \(T\).

Example:

<table>
<thead>
<tr>
<th>S:</th>
<th>ManagedBy</th>
<th>emp</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>T:</td>
<td>Emp</td>
<td>emp</td>
<td>dept</td>
</tr>
<tr>
<td>(\Sigma): st-TGD: ManagedBy(e, m) \rightarrow Emp(e, d), Dept(d, m)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\Sigma): t-TGD: Dept(d, m) \rightarrow Emp(m, d)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: no manager manages more than one department.

<table>
<thead>
<tr>
<th>T:</th>
<th>Dept</th>
<th>dept</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Sigma): t-EGD: Dept(d, m), Dept(d', m) \rightarrow d = d'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complete Example (Schema Mapping):

<table>
<thead>
<tr>
<th>S:</th>
<th>ManagedBy</th>
<th>emp</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>T:</td>
<td>Emp</td>
<td>emp</td>
<td>dept</td>
</tr>
<tr>
<td>(\Sigma): st-TGD: ManagedBy(e, m) \rightarrow Emp(e, d), Dept(d, m)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\Sigma): t-TGD: Dept(d, m) \rightarrow Emp(m, d)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\Sigma): t-EGD: Dept(d, m), Dept(d', m) \rightarrow d = d'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\*\*Example\*\*

Let \((S, T, \Sigma)\) be a schema mapping.

A source instance is an instance over \(S\).

A target instance is a \(v\)-instance \(J\) over \(T\); hence, \(J\) may have labeled nulls (or variables) instead of values.

Formally:

- We have a set \(\text{Const}\) of constants and a set \(\text{Var}\) of variables (labeled nulls).
- A source instance has only values from \(\text{Const}\).
- A target instance may have values from both \(\text{Const}\) and \(\text{Var}\).

For the sake of constraint satisfaction, we identify a target instance as an ordinary instance, interpreting each labeled null as a unique value.

This is important, since we can now define what it means for a target instance (with nulls) to satisfy a TGD/EGD.

**Definition (Solution)**

Let \(M = (S, T, \Sigma)\) be a schema mapping, and let \(I\) be a source instance. A solution is a target instance \(J\), such that \(I\) and \(J\) jointly satisfy \(\Sigma\). We denote by \(\text{Sol}_M(J)\) the set of all solutions.

**Example**

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>emp</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td>Doron</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td>Doron</td>
</tr>
<tr>
<td>Ella</td>
<td>Ahuva</td>
<td>Doron</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>emp</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td>Doron</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td>Doron</td>
</tr>
<tr>
<td>Ella</td>
<td>Ahuva</td>
<td>Doron</td>
</tr>
</tbody>
</table>

\[
\text{ManagedBy}(c, m) = \text{Emp}(c, d), \text{Dept}(d, m) \\
\text{Dept}(d, m) = \text{Emp}(m, d) \\
\text{Dept}(d, m), \text{Dept}(d', m) = d = d'
\]

**Example**

<table>
<thead>
<tr>
<th>Emp</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Chen</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Ella</td>
<td>Finance</td>
</tr>
<tr>
<td>Ahuva</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Doron</td>
<td>Finance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dept</th>
<th>emp</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>1</td>
<td>Ahuva</td>
</tr>
<tr>
<td>Chen</td>
<td>1</td>
<td>Finance</td>
</tr>
<tr>
<td>Ella</td>
<td>1</td>
<td>Ahuva</td>
</tr>
<tr>
<td>Ahuva</td>
<td>1</td>
<td>Doron</td>
</tr>
<tr>
<td>Doron</td>
<td>1</td>
<td>Finance</td>
</tr>
</tbody>
</table>
Example

ManagedBy

<table>
<thead>
<tr>
<th>emp</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>Ahuva</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
</tr>
<tr>
<td>Ella</td>
<td>Doron</td>
</tr>
</tbody>
</table>

Homomorphism

- Let \( J \) and \( K \) be two target instances
- Let \( \mu \) be a mapping from the values of \( J \) to those of \( K \)
- For a fact \( f = R(a_1, \ldots, a_k) \), we denote the fact \( R(\mu(a_1), \ldots, \mu(a_k)) \) by \( \mu(f) \)
- \( \mu \) is a homomorphism from \( J \) to \( K \) if
  - \( \mu(c) = c \) for every \( c \in \text{Const} \); and
  - \( \mu(f) \in K \) for every \( f \in J \).
- That is, a homomorphism maps every constant to itself, and every \( J \)-fact to a \( K \)-fact.

Notation

- Let \( J \) and \( K \) be two target instances
- We denote by \( J \rightarrow K \) the fact that there exists a homomorphism from \( J \) to \( K \)
- Note: many homomorphisms from \( J \) to \( K \) may exist
- \( J \rightarrow K \) means that \( K \) contains a homomorphic image of \( J \)
  - that is, a set of facts that could be obtained from \( J \) by an assignment of constants or variables to the variables

Example

Emp

<table>
<thead>
<tr>
<th>emp</th>
<th>dept</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahuva</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Barak</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Chen</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Dept

<table>
<thead>
<tr>
<th>dept</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahuva</td>
</tr>
<tr>
<td>2</td>
<td>Fady</td>
</tr>
</tbody>
</table>

Properties of Homomorphism

- The relationship \( \rightarrow \) is reflexive
  - \( J \rightarrow J \) for all \( J \)
- The relationship \( \rightarrow \) is transitive
  - For all \( J, K \) and \( L \), if \( J \rightarrow K \) and \( K \rightarrow L \) then \( J \rightarrow L \)
  - Why?
- The relationship \( \rightarrow \) is not symmetric
  - Example?
Two target instances $J$ and $K$ are \textit{isomorphic} if each one can be obtained from the other by (uniquely) renaming variables. Isomorphism is an \textit{equivalence} relation (i.e., reflexive, symmetric and transitive).

Clearly, if $J$ and $K$ are isomorphic, then $J \rightarrow K$ and $K \rightarrow J$.

But not vice versa.

Later: how to prove that this is indeed a universal solution?

As we have seen, universal solutions are not necessarily isomorphic to each other.

Moreover, different universal solutions can have different sizes.

They are, though, \textit{homomorphically equivalent} to one another.

That is, there are homomorphisms in both directions.

Variety of Universal Solutions

Let $J$ be a $v$-instance.

An \textit{endomorphism} on $J$ is a homomorphism from $J$ to itself.

The endomorphism is \textit{proper} if it maps $J$ to a proper subset of $J$.

The \textit{core} of $J$, denoted $\text{Core}(J)$, is a subinstance $J'$ of $J$ such that:

- There is a homomorphism (endomorphism) from $J$ to $J'$.
- There is no proper endomorphism on $J'$.

Why “the” core? Next...
**Theorem [HN92]**

The following hold:
- Every v-instance has a core.
- All cores of a v-instance are isomorphic.
- If $J \rightarrow K$ and $K \rightarrow J$, then $\text{Core}(J) \cong \text{Core}(K)$.

---

**Example 1 Revisited**

$R(x) \rightarrow T(x, x)$

$T(x, y), T(y, y), T(y, z) \rightarrow T(x, z)$

$I = \{ R(a) \}$

<table>
<thead>
<tr>
<th>$T$</th>
<th>$\Rightarrow$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$a_1 \rightarrow a_1$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$a_2 \rightarrow a_2$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$a_3 \rightarrow a_3$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$a_4 \rightarrow a_4$</td>
<td>$a_4$</td>
</tr>
</tbody>
</table>

---

**Example 2**

<table>
<thead>
<tr>
<th>$T$</th>
<th>$\Rightarrow$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$a_1 \rightarrow a$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$a_2 \rightarrow a$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$a_3 \rightarrow a$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$a_4 \rightarrow a$</td>
<td>$a_4$</td>
</tr>
</tbody>
</table>

---

**Endomorphic Images of Solutions**

- Let $(S, T, \Sigma)$ be a schema mapping, and let $I$ be a source instance.
- Suppose that $J$ is a solution and $\mu$ is an endomorphism over $J$.
- Is $\mu(J)$ necessarily a solution?
- No!

---

**Cores of Solutions Are Solutions**

**Theorem [FKP05]**

Let $(S, T, \Sigma)$ be a schema mapping, let $I$ be a source instance, and let $J$ be target instance. If $J$ is a solution, then $\text{Core}(J)$ is a solution.
• We now get a good candidate for a solution: the core of a universal solution
  • Simply called a core solution
• We know:
  • There is a universal solution if and only if there is a core solution
  • All core solutions are isomorphic
  • Core solutions are universal solutions
    • why?
    • Core solutions are the smallest universal solutions
    • why?

We evaluate a query \( Q \) over a target instance \( J \) by treating each labeled null as a unique value

Clearly, different solutions \( J \in \text{Sol}_M(I) \) may have different results \( Q(J) \)

**Certain Answers**

**Definition (Certain Answers)**

Let \( M \) be a schema mapping, \( I \) a source instance, and \( Q \) be a target query. We denote by \( \text{Certain}_Q, M(I) \) the relation that consists of all the tuples that occur in \( Q(J) \) for all solutions \( J \); that is:

\[
\text{Certain}_Q, M(I) = \bigcap \{ Q(J) \mid J \in \text{Sol}_M(I) \}
\]

A tuple in \( \text{Certain}_Q, M(I) \) is called a certain answer.

Recall that a conjunctive query (CQ) is a query of the form

\[
Q(x) = \exists y \left[ \psi_1(x, y) \land \cdots \land \psi_k(x, y) \right]
\]

where each \( \psi_i(x, y) \) is an atomic query

• A union of conjunctive queries (UCQ) is a query of the form

\[
Q(x) = \psi_1(x) \lor \cdots \lor \psi_k(x)
\]

where each \( Q(x) = \psi_i(x) \) is a CQ

If \( Q \) is a target query and \( J \) is a target instance, then \( Q(J) \)

denotes the relation obtained from \( Q(J) \) by removing every tuple with one or more nulls

**Answering UCQs via Universal Solutions**

**Theorem (FKMP'03)**

Let \( M = (S, T, \Sigma) \) be a schema mapping, \( Q \) a target UCQ, and \( I \) a source instance. If \( J \) is a universal solution, then

\[
Q(J)_I = \text{Certain}_Q, M(I)
\]
The Chase is a standard technique in reasoning about databases with integrity constraints [BV84].

- Idea:
  - Start with the original database instance
  - Repeatedly, look for a violation of a constraint, and fix by revising the instance
  - . . . until no violations remain (or the violation cannot be fixed)

### Chase Steps for Data Exchange

- Start with $J = \emptyset$
- TGD violation: $\psi(x) \Rightarrow \exists y \psi(x, y)$
- **Violation**: an assignment $\alpha$ to $x$, such that $\psi(\alpha(x))$ holds, but there is no $b$ such that $\psi(\alpha(x), b)$
- **Fix**: add the facts of $\psi(\alpha(x), y)$, where each existential variable in $y$ is replaced with a fresh null
- EGD violation: $\psi(x) \Rightarrow y_1 = y_2$
- **Violation**: An assignment $\alpha$ to $x$, such that $\alpha(\psi(x))$ holds, but $\alpha(y_1) \neq \alpha(y_2)$
- **Fix**: 3 cases
  - $\alpha(y_1)$ is a labeled null: replace every $\alpha(y_2)$ with $\alpha(y_1)$ in $J$
  - $\alpha(y_2)$ is a labeled null: replace every $\alpha(y_1)$ with $\alpha(y_2)$ in $J$
  - Both $\alpha(y_1)$ and $\alpha(y_2)$ are constants: terminate with failure

### Chase Example

- **ManagedBy**
- **Emp**
- **Dept**

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>emp</td>
<td>mgr</td>
</tr>
<tr>
<td>Chen</td>
<td>emp</td>
<td>dept</td>
</tr>
<tr>
<td>Ella</td>
<td>emp</td>
<td>dept</td>
</tr>
<tr>
<td>Ahuva</td>
<td>dept</td>
<td>mgr</td>
</tr>
<tr>
<td>Doron</td>
<td>dept</td>
<td>mgr</td>
</tr>
</tbody>
</table>

- ManagedBy($e, m$) $\Rightarrow$ Emp($e, d$), Dept($d, m$)
- Dept($d, m$), Dept($d', m$) $\Rightarrow$ $d = d'$
- Dept($d, m$) $\Rightarrow$ Emp($m, d$)
### Chase Example

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp</td>
<td>mgr</td>
<td>emp</td>
</tr>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td>Barak</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td>Chen</td>
</tr>
<tr>
<td>Ella</td>
<td>Doron</td>
<td>Ella</td>
</tr>
</tbody>
</table>

- ManagedBy(e, m) → Emp(e, d), Dept(d, m)
- Dept(d, m), Dept(d', m) → d = d'
- Dept(d, m) → Emp(m, d)

### Chase Example

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp</td>
<td>mgr</td>
<td>emp</td>
</tr>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td>Barak</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td>Chen</td>
</tr>
<tr>
<td>Ella</td>
<td>Doron</td>
<td>Ella</td>
</tr>
</tbody>
</table>

- ManagedBy(e, m) → Emp(e, d), Dept(d, m)
- Dept(d, m), Dept(d', m) → d = d'
- Dept(d, m) → Emp(m, d)

---

### Chase Example

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp</td>
<td>mgr</td>
<td>emp</td>
</tr>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td>Barak</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td>Chen</td>
</tr>
<tr>
<td>Ella</td>
<td>Doron</td>
<td>Ella</td>
</tr>
</tbody>
</table>

- ManagedBy(e, m) → Emp(e, d), Dept(d, m)
- Dept(d, m), Dept(d', m) → d = d'
- Dept(d, m) → Emp(m, d)

### Chase Example

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp</td>
<td>mgr</td>
<td>emp</td>
</tr>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td>Barak</td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td>Chen</td>
</tr>
<tr>
<td>Ella</td>
<td>Doron</td>
<td>Ella</td>
</tr>
</tbody>
</table>

- ManagedBy(e, m) → Emp(e, d), Dept(d, m)
- Dept(d, m), Dept(d', m) → d = d'
- Dept(d, m) → Emp(m, d)
Chase Gives a Universal Solution

**Theorem [FKMP05]**

Let $M = (S, T, \Sigma)$ be a schema mapping and $I$ a source instance.

- If the chase terminates successfully, then the result is a universal solution.
- If the chase terminates with failure, then there is no solution at all.

The result of a successful chase is called a **canonical** universal solution.

Chase Example

<table>
<thead>
<tr>
<th>ManagedBy</th>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp</td>
<td>mgr</td>
<td></td>
</tr>
<tr>
<td>Barak</td>
<td>Ahuva</td>
<td></td>
</tr>
<tr>
<td>Chen</td>
<td>Ahuva</td>
<td></td>
</tr>
<tr>
<td>Ella</td>
<td>Doron</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp</th>
<th>dept</th>
<th>dept</th>
<th>mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barak</td>
<td>1_1</td>
<td>Ahuva</td>
<td></td>
</tr>
<tr>
<td>Chen</td>
<td>1_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ella</td>
<td>1_3</td>
<td>Doron</td>
<td></td>
</tr>
<tr>
<td>Ahuva</td>
<td>1_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doron</td>
<td>1_3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ManagedBy($e, m$) $\rightarrow$ Emp($e, d$), Dept($d, m$)
- Dept($d, m$), Dept($d', m$) $\rightarrow$ $d = d'$
- Dept($d, m$) $\rightarrow$ Emp($m, d$)

Chase Failure and Termination

There are three kinds of chase executions:

- Termination with success
- Termination with failure
- Non-termination
  - How can that be?
  - Person($x$) $\rightarrow$ ChildOf($x, y$)  ChildOf($x, y$) $\rightarrow$ ChildOf($y, z$)

Chase Termination

- We would like to be able to test whether a schema mapping $M = (S, T, \Sigma)$ is such that the chase is guaranteed to terminate
- Unfortunately, this is provably impossible
The following problems are undecidable. Given a schema mapping \((S,T,\Sigma)\) and a source instance \(I\):

- Is there any solution?
- Is there any universal solution?
- Is there any terminating chase?
- Is every chase terminating?

**Theorem** [KPT06, DNR08]

Theorem: **Weak acyclicity** is a property of a schema mapping that guarantees termination of the chase (and further consequent goodness properties).

**Dependency Graph**

- Let \((S,T,\Sigma)\) be a schema mapping
- A **position** is an expression of the form \(R.A\), where \(R\) is a relation name and \(A\) is an attribute of \(R\)
- The **dependency graph** of \(\Sigma\) is the directed graph \(G_\Sigma\) defined as follows:
  - The nodes are the positions
  - For each t-TGD \(\varphi(x) \rightarrow \exists y \psi(x,y)\), position \(p\) at \(\varphi(x)\), position \(q\) at \(\psi(x,y)\) there is an edge \(p \rightarrow q\) if:
    - \(p\) and \(q\) have the same variable (ordinary edge)
    - \(q\) has an existential variable (special edge)

**Example**

- ManagedBy\((e, m)\) \(\rightarrow\) Emp\((e, d)\), Dept\((d, m)\)
- Dept\((d, m)\) \(\rightarrow\) Emp\((m, d)\)
- Emp\((e, d)\) \(\rightarrow\) Dept\((d, m)\)

\[
\text{Dept.dept} \rightarrow \text{Emp.dept} \quad \text{Dept.mgr} \rightarrow \text{Emp.emp}
\]

**Weak Acyclicity**

- \(\Sigma\) is **weakly acyclic** if \(G_\Sigma\) does not have any cycle that goes through a special edge
- Note: weak acyclicity is a property of the t-TGDs (independent of the st-TGDs and t-EGDs)
- Weak acyclicity can be tested in polynomial time in \(\Sigma\)

**Example: Weak Acyclic**

- ManagedBy\((e, m)\) \(\rightarrow\) Emp\((e, d)\), Dept\((d, m)\)
- Dept\((d, m)\) \(\rightarrow\) Emp\((m, d)\)

\[
\text{Dept.dept} \rightarrow \text{Emp.dept} \quad \text{Dept.mgr} \rightarrow \text{Emp.emp}
\]
### Example: Not Weakly Acyclic

- ManagedBy(e, m) → Emp(e, d), Dept(d, m)
- Dept(d, m) → Emp(m, d)
- Emp(e, d) → Dept(d, m)

\[
\begin{align*}
\text{Dept.dept} & \quad \rightarrow \quad \text{Emp.dept} \\
\text{Dept.mgr} & \quad \rightarrow \quad \text{Emp.emp}
\end{align*}
\]

### Weak Acyclicity and Chase Termination

**Theorem [FKMP05]**

Let \( M = (S, T, \Sigma) \) be a schema mapping. Assume that \( \Sigma \) is weakly acyclic. Given a source instance \( I \), the chase terminates in polynomial time in \( |I| \).

(The degree of the polynomial is determined by \( M \).)

### Algorithm for Answering Target UCQs

- We now have an algorithm for computing the certain answers, given a source instance \( I \) for a schema mapping \((S, T, \Sigma)\):
  - chase \( I \) and \( \Sigma \)
  - if the chase fails then abort(“no solution”)
  - else let \( J \) be the resulting solution
  - return \( Q(J) \)

- Of course, the algorithm assumes that the chase terminates
- If \( \Sigma \) is weakly acyclic, we get a polynomial-time algorithm

### Computing a Core Solution

- Can we construct a core solution in polynomial time if \( \Sigma \) is weakly acyclic?
- The straightforward way is:
  - Compute a universal solution \( J \) via chase
  - Compute the core of \( J \)
- Does it return the correct result?
- Is it a polynomial-time strategy?
- What is the complexity of finding the core of a \( v \)-instance?

### Reduction from 3-Colorability (1)

- We show that it is NP-hard to compute the core of a \( v \)-instance
- Reduction from 3-colorability: given a graph \( g \), is there a legal node coloring by 3 colors?
- Denote \( g = (V, E) \) where \( V = \{v_1, \ldots, v_n\} \)
- Construct a \( v \)-instance \( D \) with a single relation \( R_D \) that consists of only labeled nulls
  - \( R(i, i) \) for all \( i < j \) such that \( \{v_i, v_j\} \) is an edge
  - \( R(x, y) \) for every \( x \neq y \) in \( \{r, g, b\} \)
  - Denote this part of \( D \) as \( D_A \)
Reduction from 3-Colorability (2)

- Claim: the core of $D$ is $D_{\Delta}$ if and only if $g$ is 3-colorable
- Proof steps:
  - If $Core(D) = D_{\Delta}$ then the endomorphism is a legal 3-coloring
  - A 3-coloring defines an endomorphism to $D_{\Delta}$
  - $Core(D_{\Delta}) = D_{\Delta}$
- This establishes NP-hardness
- Nevertheless, $D$ is not necessarily the result of chasing with a weakly acyclic $\Sigma$!
- Gottlob and Nash [GN08] proved that, if $J$ is the result of such a chase, then $Core(D)$ can, in fact, be computed in polynomial time.

**Theorem [GN08]**

Let $\mathcal{M} = (S, T, \Sigma)$ be a schema mapping. Assume that $\Sigma$ is weakly acyclic. Given a source instance $I$, a core solution can be found in polynomial time, if any solution exists.

References