Analyzing, Comparing and Debugging Schema Mappings

Emanuel Sallinger

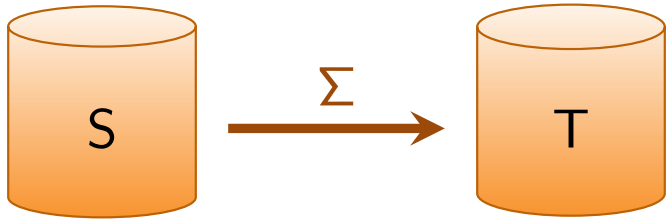
Vienna University of Technology
Institute of Information Systems
Database and Artificial Intelligence Group

DEIS'10

11 November, 2010

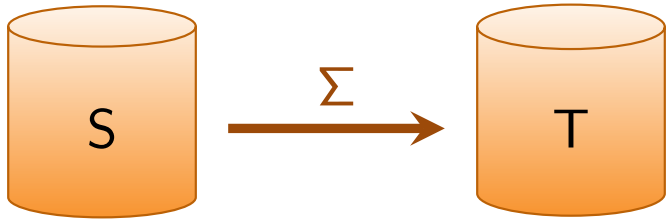
Outline

Given a schema mapping $\mathcal{M}$ ...



Outline

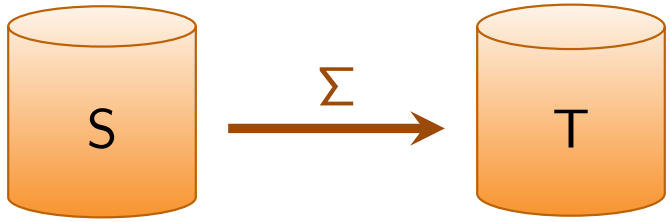
Given a schema mapping $\mathcal{M}$ ...



- What does it **do**? (analyzing)
- Are there any **errors** in it? (debugging)

Outline

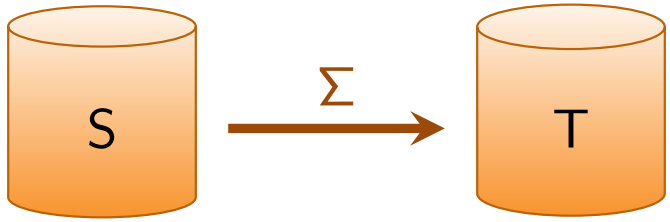
Given a schema mapping $\mathcal{M}$ ...



- What does it **do**? (analyzing)
- Are there any **errors** in it? (debugging)
- Is there a **better** one ... (comparing/optimizing)

Outline

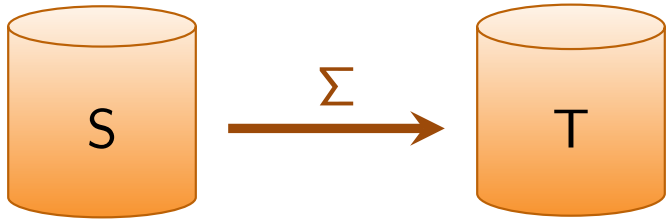
Given a schema mapping $\mathcal{M}$ ...



- What does it **do**? (analyzing)
- Are there any **errors** in it? (debugging)
- Is there a **better** one ... (comparing/optimizing)
  - ...that is **equivalent**?

Outline

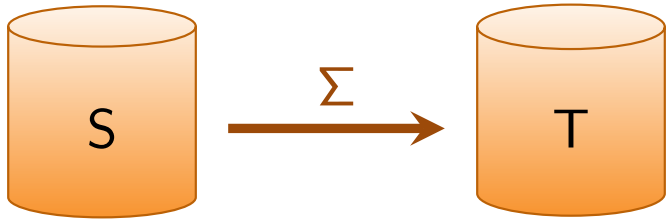
Given a schema mapping $\mathcal{M}$ ...



- What does it **do**? (analyzing)
- Are there any **errors** in it? (debugging)
- Is there a **better** one ... (comparing/optimizing)
  - ...that is **equivalent**?
  - ...that is equivalent for **specific purposes**, e.g. data exchange?

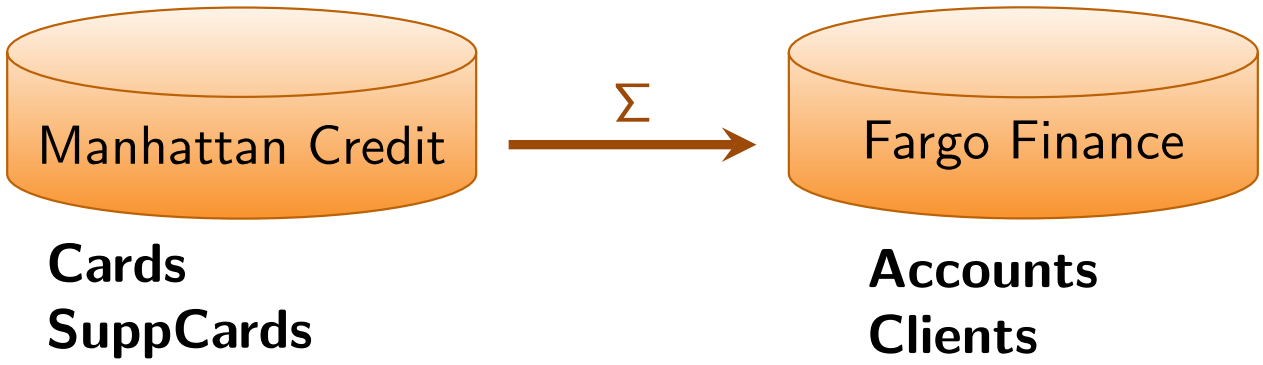
Outline

Given a schema mapping $\mathcal{M}$ ...

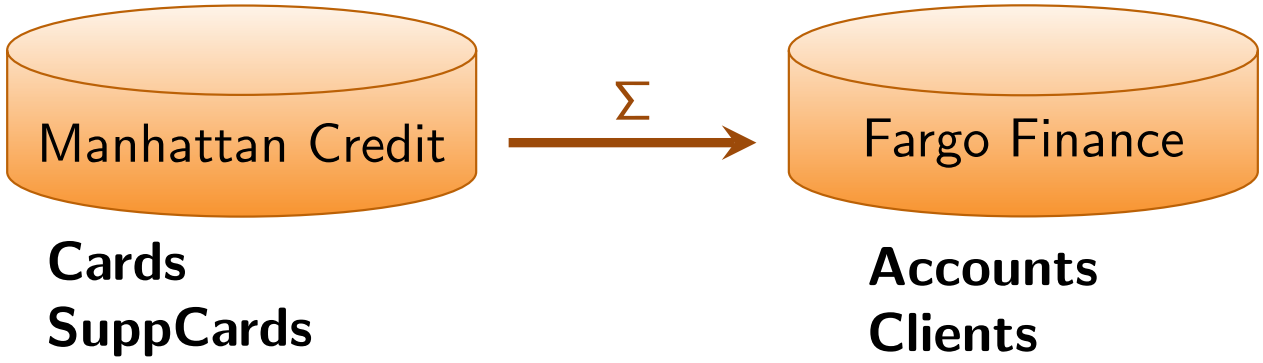


- What does it **do**? (analyzing)
- Are there any **errors** in it? (debugging)
- Is there a **better** one ... (comparing/optimizing)
  - ...that is **equivalent**?
  - ...that is equivalent for **specific purposes**, e.g. data exchange?
  - What about other comparison criteria?

Debugging with Routes



Debugging with Routes



σ_1: $\mathbf{Cards}(cn, l, s, n, m, sal, loc) \rightarrow$
$\exists A\,(\mathbf{Accounts}(cn, l, s) \wedge \mathbf{Clients}(s, m, m, sal, A))$

σ_2: $\mathbf{SuppCards}(an, s, n, a) \rightarrow \exists M, I\ \mathbf{Clients}(s, n, M, I, a)$

σ_3: $\mathbf{Accounts}(a, l, s) \rightarrow \exists N, M, I, A\ \mathbf{Clients}(s, N, M, I, A)$

σ_4: $\mathbf{Clients}(s, n, m, i, a) \rightarrow \exists N, L\ \mathbf{Accounts}(N, L, s)$

σ_5: $\mathbf{Accounts}(a, l, s) \wedge \mathbf{Accounts}(a', l', s) \rightarrow l = l'$

Debugging with Routes

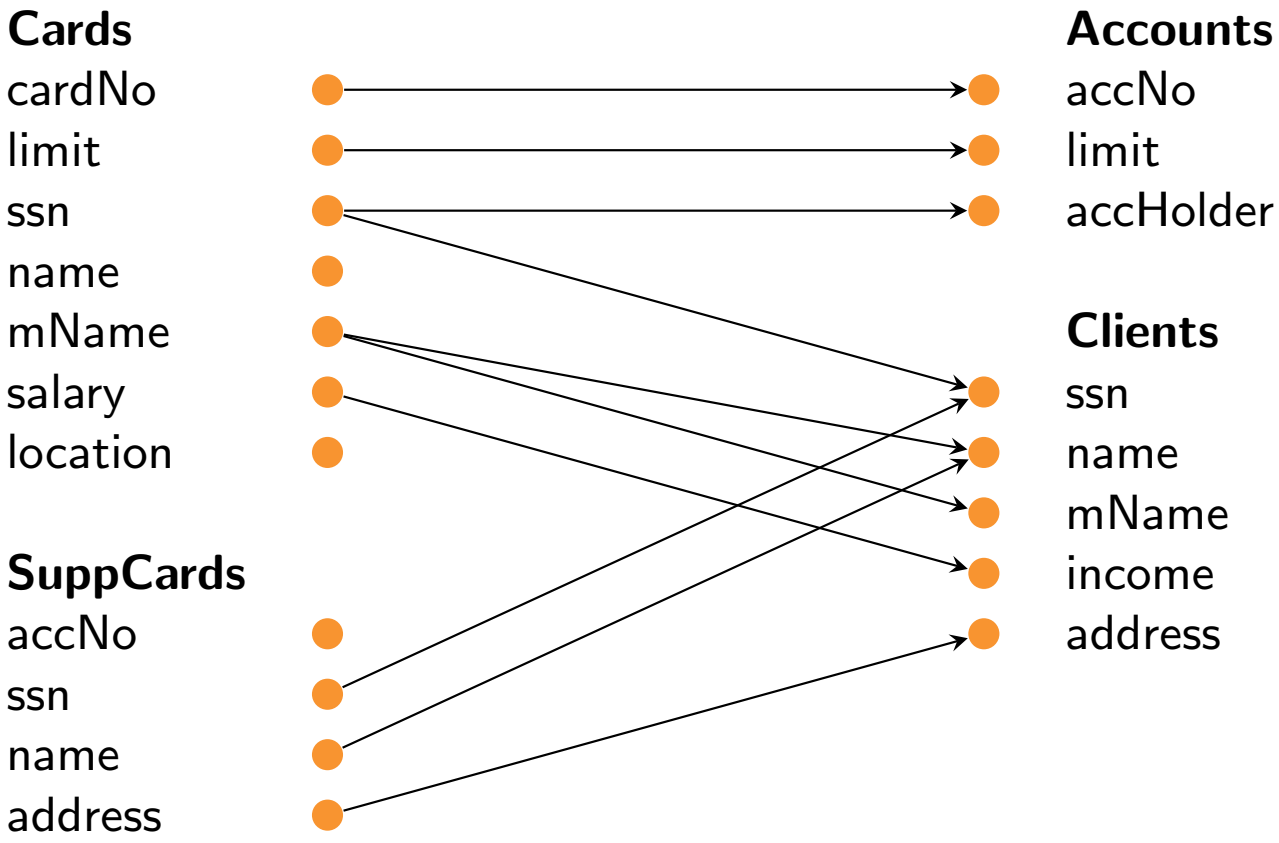
Cards
- cardNo
- limit
- ssn
- name
- mName
- salary
- location

SuppCards
- accNo
- ssn
- name
- address

Accounts
- accNo
- limit
- accHolder

Clients
- ssn
- name
- mName
- income
- address

Debugging with Routes



σ_1: $\mathbf{Cards}(cn, l, s, n, m, sal, loc) \rightarrow$
$\exists A\,(\mathbf{Accounts}(cn, l, s) \wedge \mathbf{Clients}(s, m, m, sal, A)$

σ_2: $\mathbf{SuppCards}(an, s, n, a) \rightarrow \exists M, I\ \mathbf{Clients}(s, n, M, I, a)$

Debugging with Routes

s_1, s_2

s_1: **Cards**(6689, 15K, 434, J.Long, Smith, 50K, Seattle)

s_2: **SuppCards**(6689, 234, A.Long, California)

Debugging with Routes

s_1, s_2 $\qquad\qquad$ t_1, t_2, t_3, t_4

s_1: **Cards**(6689, 15K, 434, J.Long, Smith, 50K, Seattle)

s_2: **SuppCards**(6689, 234, A.Long, California)

t_1: **Accounts**(6689, 15K, 434)

t_2: **Accounts**(N_1, 50K, 234)

t_3: **Clients**(434, Smith, Smith, 50K, A_1)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

Debugging with Routes

s_1, s_2 $\qquad\qquad$ t_1, t_2, t_3, t_4

s_1: **Cards**(6689, 15K, 434, J.Long, Smith, 50K, Seattle)

s_2: **SuppCards**(6689, 234, A.Long, California)

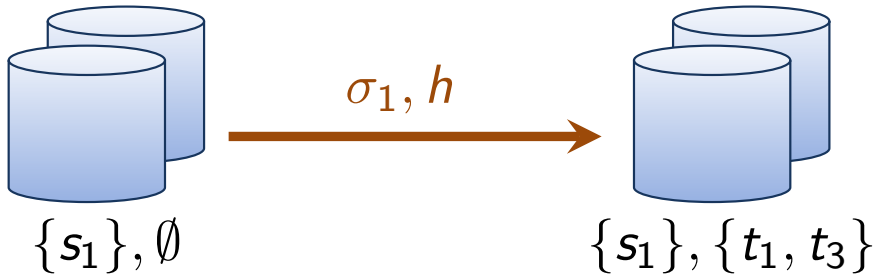
t_1: **Accounts**(6689, 15K, 434)

t_2: **Accounts**(N_1, 50K, 234)

t_3: **Clients**(434, Smith, Smith, 50K, A_1)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

Debugging with Routes



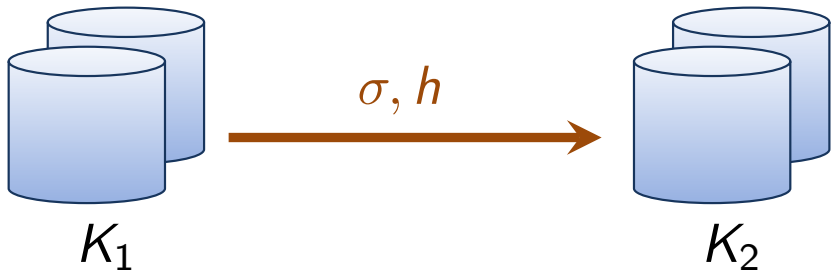
s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

σ_1: **Cards**$(cn, l, s, n, m, sal, loc) \rightarrow$
$\exists A\,($**Accounts**$(cn, l, s) \wedge$ **Clients**(s, m, m, sal, A)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434, n \mapsto$ J.Long, $m \mapsto$ Smith,
$sal \mapsto 50K, loc \mapsto$ Seattle, $A \mapsto A_1\}$

t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

Debugging with Routes



Definition [CT06]

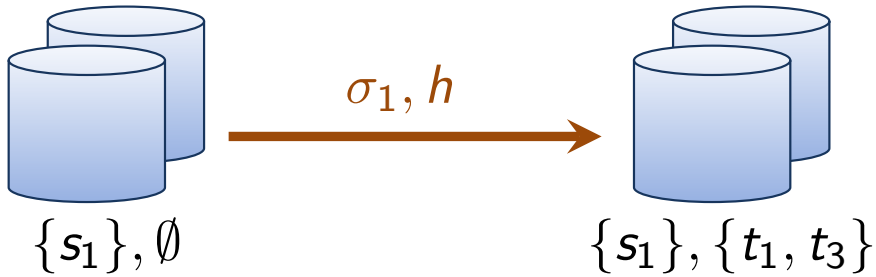
A **satisfaction step** is given as

$$K_1 \xrightarrow{\sigma,h} K_2$$

- K_1 is an instance such that $K_1 \subseteq K$ and K satisfies σ
- σ is a tgd $\varphi(\vec{x}) \rightarrow \exists \vec{y}\, \psi(\vec{x}, \vec{y})$
- h is a homomorphism from $\varphi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to K such that h is also a homomorphism from $\varphi(\vec{x})$ to K_1
- K_2 is the **result** of satisfying σ on K_1 with homomorphism h, where

$$K_2 = K_1 \cup h(\psi(\vec{x}, \vec{y}))$$

Debugging with Routes



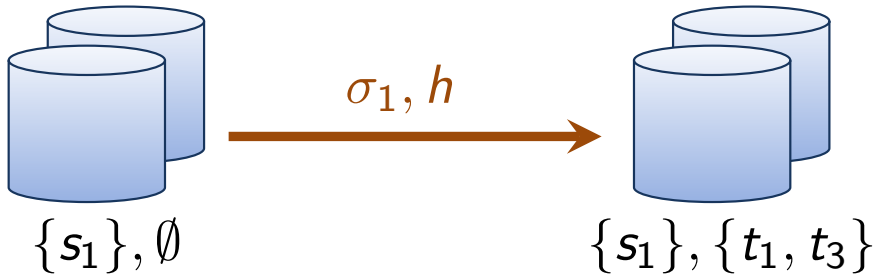
s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

σ_1: **Cards**$(cn, l, s, n, m, sal, loc) \rightarrow$
$\exists A\,($**Accounts**$(cn, l, s) \wedge$ **Clients**(s, m, m, sal, A)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434, n \mapsto$ J.Long, $m \mapsto$ Smith,
$sal \mapsto 50K, loc \mapsto$ Seattle, $A \mapsto A_1\}$

t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

Debugging with Routes



s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

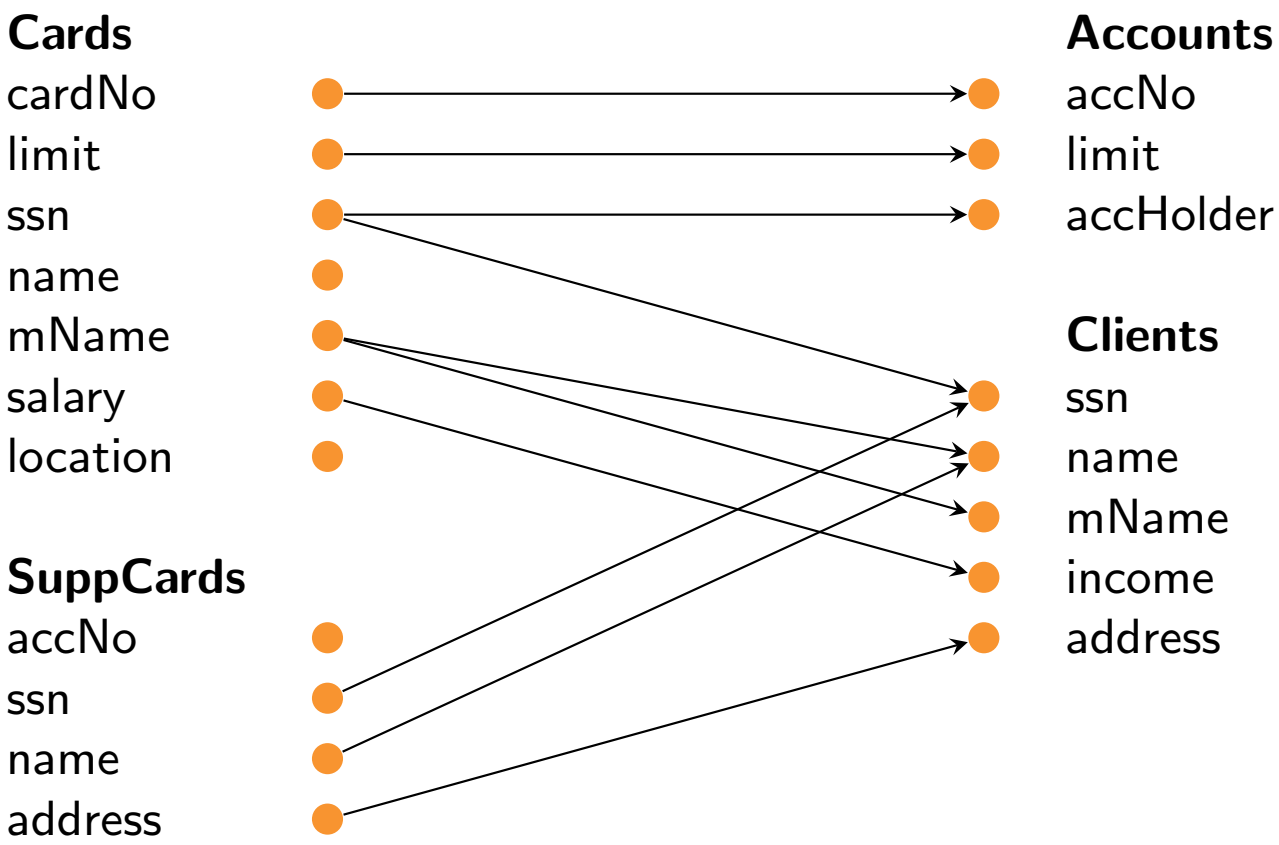
σ_1: **Cards**$(cn, l, s, n, m, sal, loc) \rightarrow$
$\exists A\,(\textbf{Accounts}(cn, l, s) \wedge \textbf{Clients}(s, n, m, sal, A)$

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434, n \mapsto$ J.Long, $m \mapsto$ Smith,
$sal \mapsto 50K, loc \mapsto$ Seattle, $A \mapsto A_1\}$

t_1: **Accounts**(6689, 15K, 434)

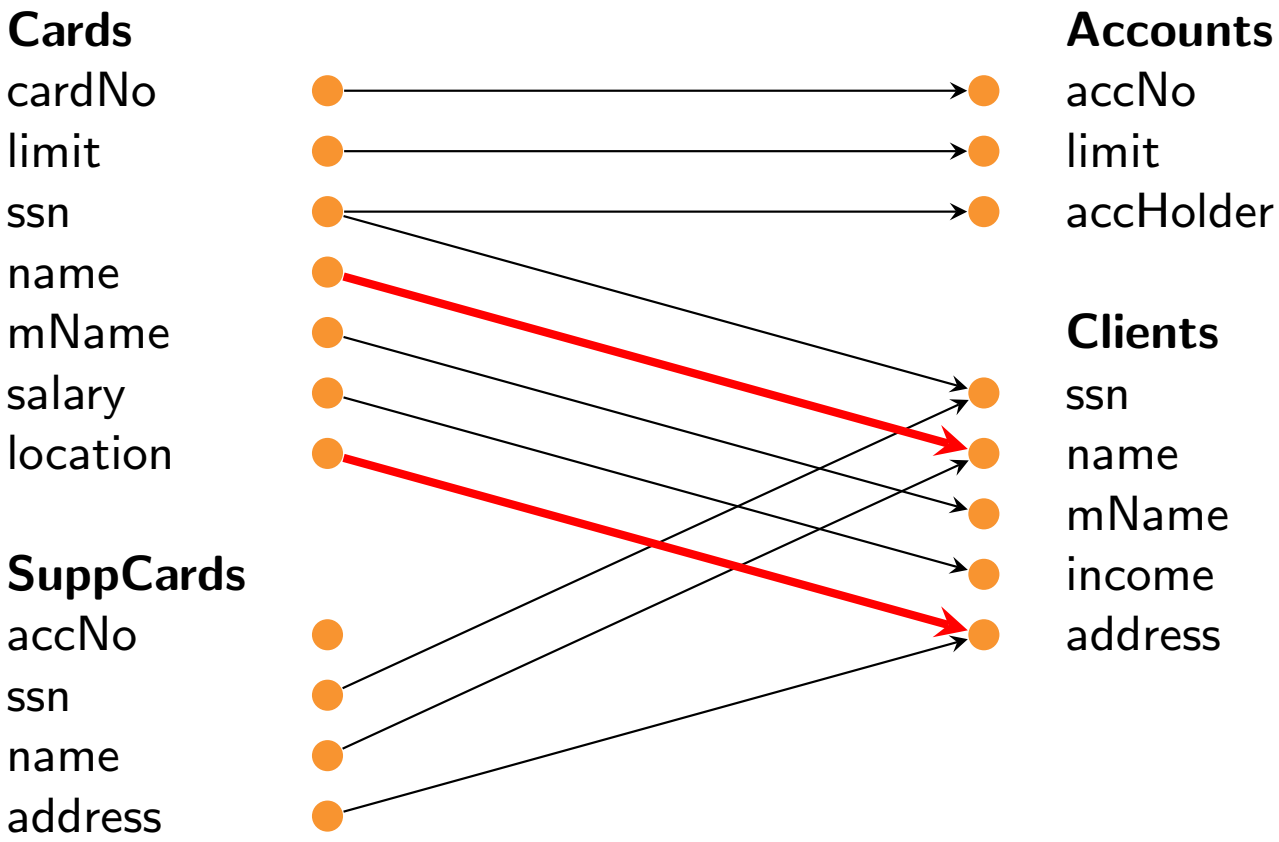
t_3: **Clients**(434, Smith, Smith, 50K, A_1)

Debugging with Routes



σ_1: **Cards**$(cn, l, s, n, m, sal, loc) \rightarrow$
$\exists A\,($**Accounts**$(cn, l, s) \wedge$ **Clients**(s, m, m, sal, A)

Debugging with Routes



σ_1': **Cards**$(cn, l, s, n, m, sal, loc) \rightarrow$
(**Accounts**$(cn, l, s) \wedge$ **Clients**(s, n, m, sal, loc))

Debugging with Routes

s_1, s_2 $\qquad\qquad$ t_1, t_2, t_3, t_4

s_1: **Cards**(6689, 15K, 434, J.Long, Smith, 50K, Seattle)

s_2: **SuppCards**(6689, 234, A.Long, California)

t_1: **Accounts**(6689, 15K, 434)

t_2: **Accounts**(N_1, 50K, 234)

t_3: **Clients**(434, Smith, Smith, 50K, A_1)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

Debugging with Routes

s_1, s_2 $\qquad\qquad$ t_1, t_2, t_3, t_4

s_1: **Cards**(6689, 15K, 434, J.Long, Smith, 50K, Seattle)

s_2: **SuppCards**(6689, 234, A.Long, California)

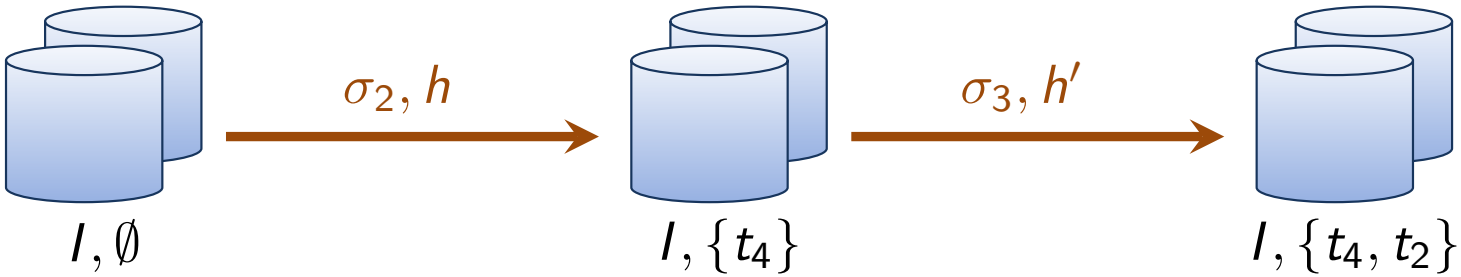
t_1: **Accounts**(6689, 15K, 434)

t_2: **Accounts**(N_1, 50K, 234)

t_3: **Clients**(434, Smith, Smith, 50K, A_1)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

Debugging with Routes



s_2: **SuppCards**(6689, 234, A.Long, California)

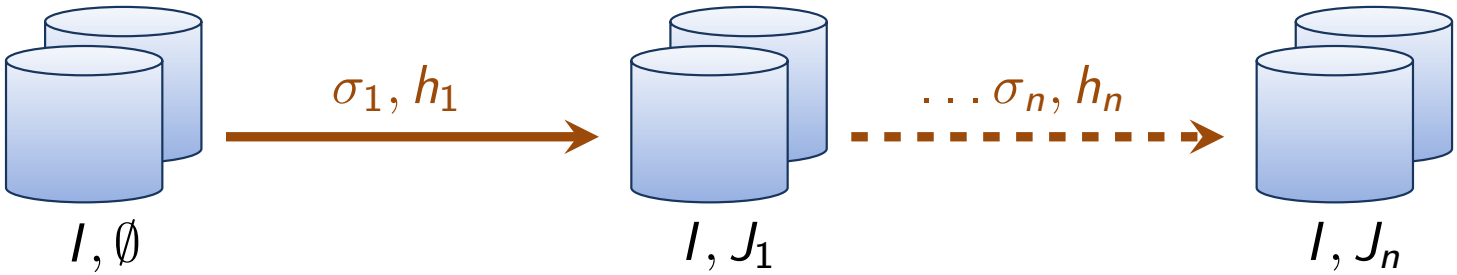
σ_2: **SuppCards**$(an, s, n, a) \rightarrow \exists M, I$ **Clients**(s, n, M, I, a)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

σ_4: **Clients**$(s, n, m, i, a) \rightarrow \exists N, L$ **Accounts**(N, L, s)

t_2: **Accounts**$(N_1, 50K, 234)$

Debugging with Routes



Definition [CT06]

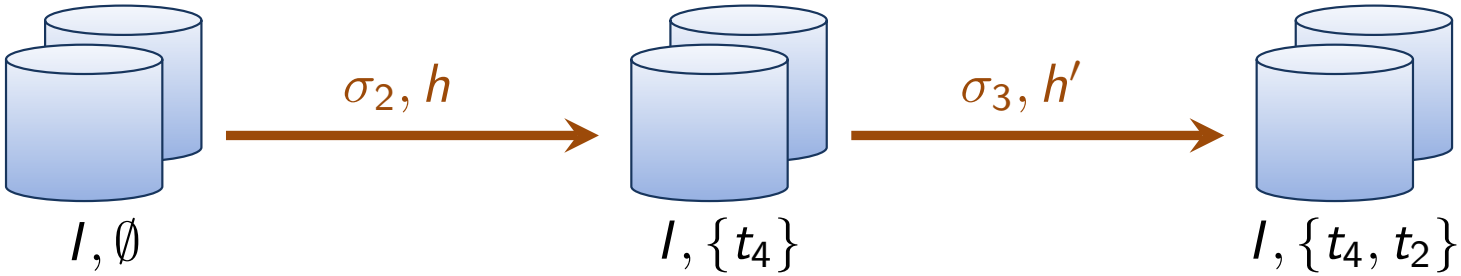
A **route** for J_s with $\mathcal{M}$, I and J is a sequence of satisfaction steps

$$(I, \emptyset) \xrightarrow{\sigma_1, h_1} (I, J_1) \ldots \xrightarrow{\sigma_n, h_n} (I, J_n)$$

where

- J is a solution of I under $\mathcal{M}$
- $J_i \subseteq J$ and σ_i are from $\mathcal{M}$
- $J_s \subseteq J_n$

Debugging with Routes



s_2: **SuppCards**(6689, 234, A.Long, California)

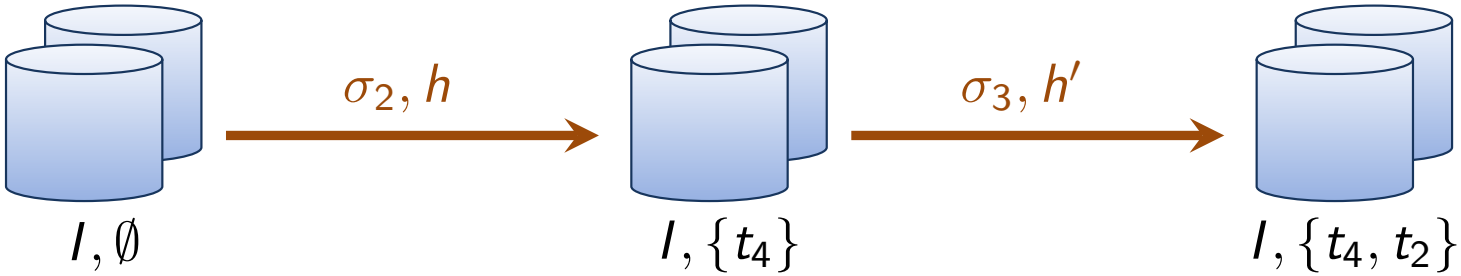
σ_2: **SuppCards**$(an, s, n, a) \rightarrow \exists M, I$ **Clients**(s, n, M, I, a)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

σ_4: **Clients**$(s, n, m, i, a) \rightarrow \exists N, L$ **Accounts**(N, L, s)

t_2: **Accounts**$(N_1$, 50K, 234)

Debugging with Routes



s_2: **SuppCards**(6689, 234, A.Long, California)

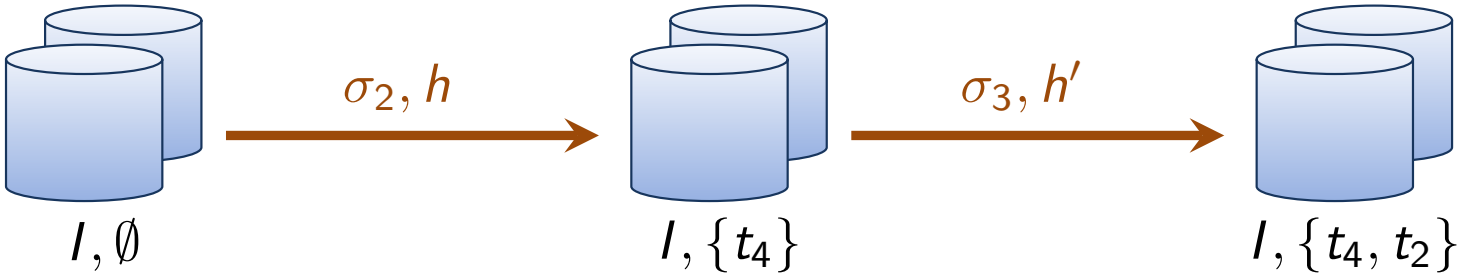
σ_2: **SuppCards**$(an, s, n, a) \rightarrow \exists M, I$ **Clients**(s, n, M, I, a)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

σ_4: **Clients**$(s, n, m, i, a) \rightarrow \exists N, L$ **Accounts**(N, L, s)

t_2: **Accounts**(N_1, 50K, 234)

Debugging with Routes



s_2: **SuppCards**(6689, 234, A.Long, California)

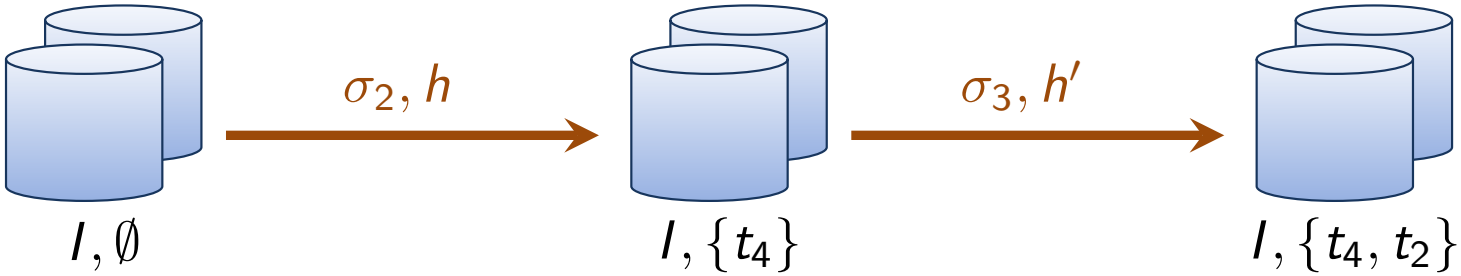
σ_2: **SuppCards**$(an, s, n, a) \rightarrow \exists M, I$ **Clients**(s, n, M, I, a)

t_4: **Clients**(234, A.Long, M_1, I_1, California)

σ_4: **Clients**$(s, n, m, i, a) \rightarrow \exists N, L$ **Accounts**(N, L, s)

t_2: **Accounts**(N_1, 50K, 234)

Debugging with Routes



s_1: **Cards**(6689, 15K, 434, J.Long, Smith, 50K, Seattle)

s_2: **SuppCards**(6689, 234, A.Long, California)

σ_2': $\mathbf{Cards}(cn, l, s_1, n_1, m, sal, loc) \wedge \mathbf{SuppCards}(cn, s_2, n_2, a) \rightarrow$
$\exists M, I\,(\mathbf{Clients}(s_2, n_2, M, I, a) \wedge \mathbf{Accounts}(cn, l, s_2)$

t_2': **Accounts**(6689, 15K, 234)

Computing Routes

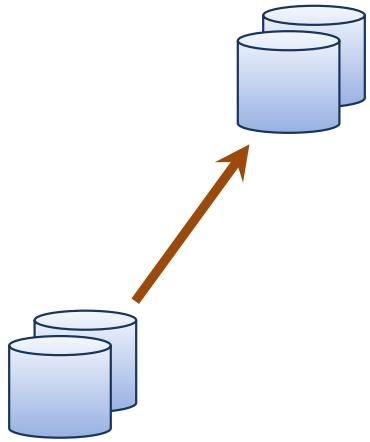
In general, a single route is not sufficient for analyzing and debugging.

Computing Routes

In general, a single route is not sufficient for analyzing and debugging.

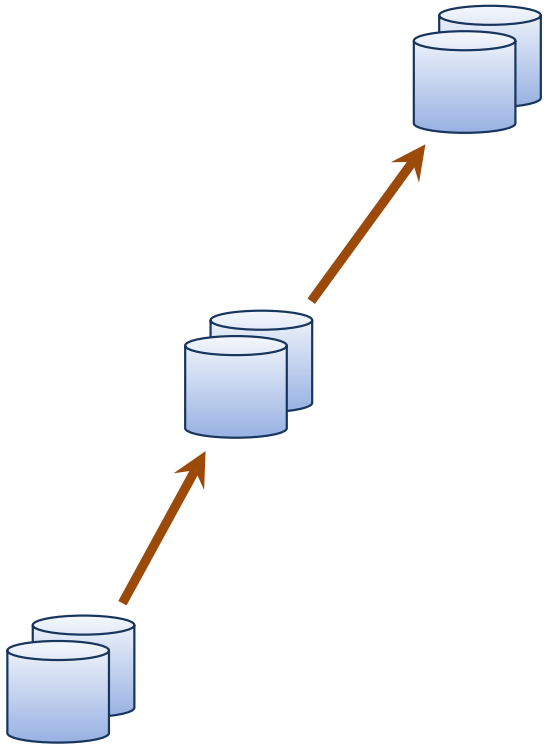
Computing Routes

In general, a single route is not sufficient for analyzing and debugging.



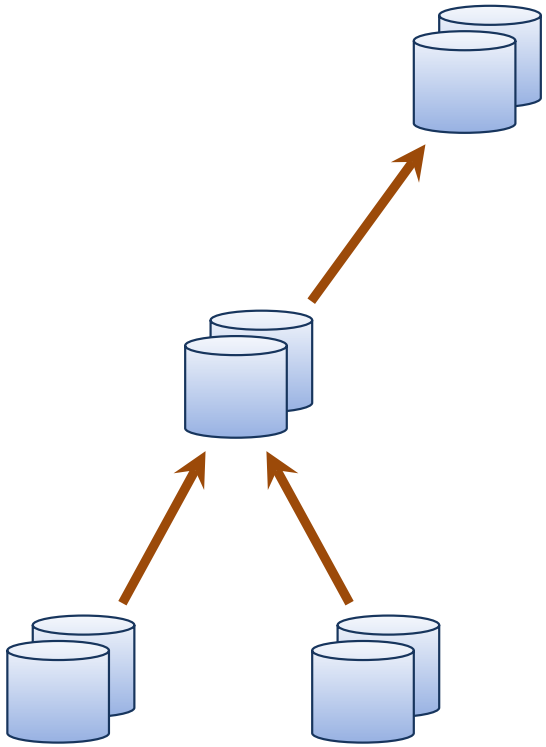
Computing Routes

In general, a single route is not sufficient for analyzing and debugging.



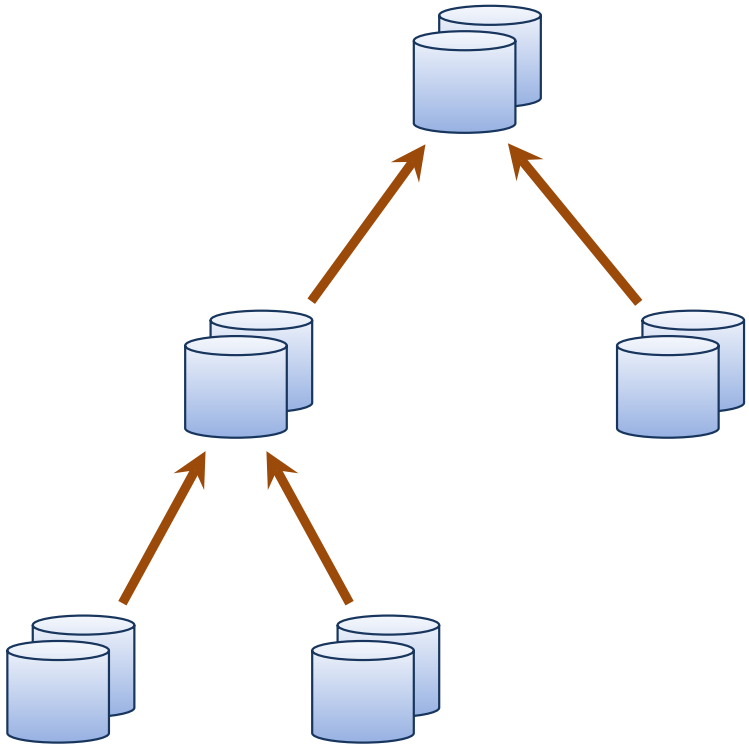
Computing Routes

In general, a single route is not sufficient for analyzing and debugging.



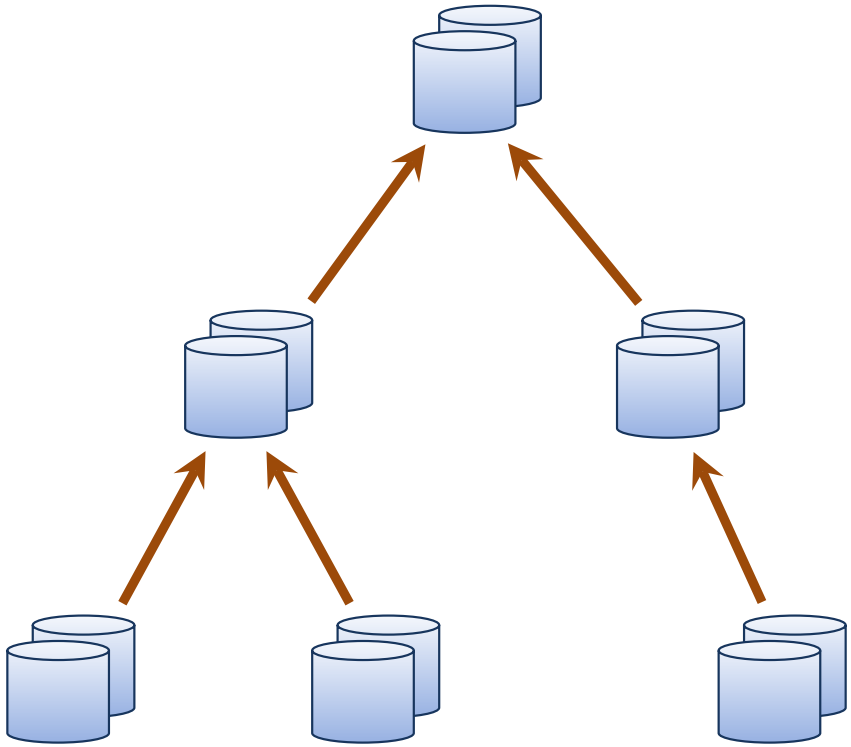
Computing Routes

In general, a single route is not sufficient for analyzing and debugging.

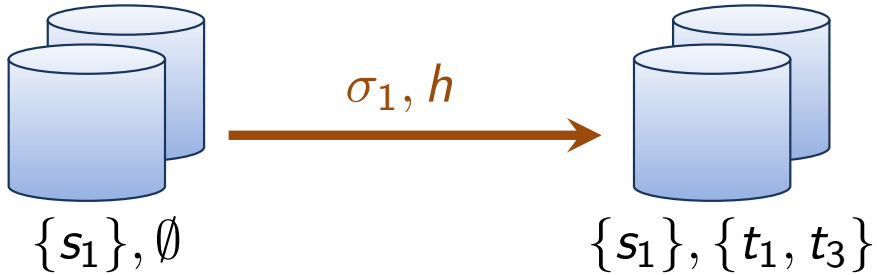


Computing Routes

In general, a single route is not sufficient for analyzing and debugging.



Computing Routes



s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

Computing Routes

s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

t_1: **Accounts**(6689, 15K, 434)

t_3: **Clients**(434, Smith, Smith, 50K, A_1)

h:

Computing Routes

s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434,$

1. Map an atom from $\psi(\vec{x}, \vec{y})$ to t. Add it to h.

Computing Routes

s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)

σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

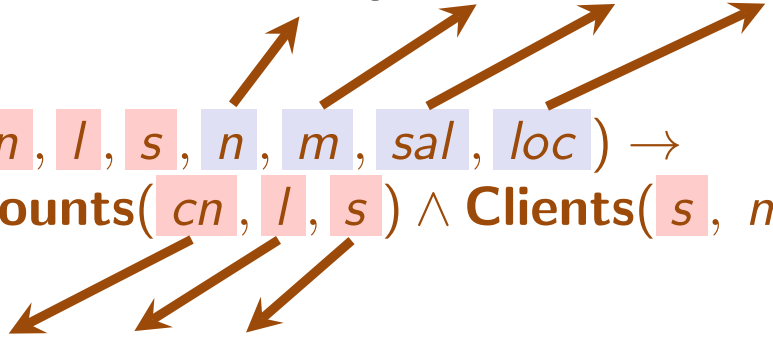
t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434,$

1. Map an atom from $\psi(\vec{x}, \vec{y})$ to t. Add it to h.

Computing Routes

s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)



σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

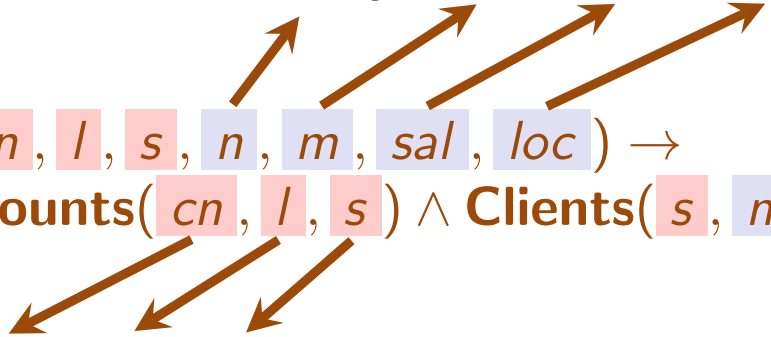
t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434,$
$n \mapsto$ J.Long, $m \mapsto$ Smith, $sal \mapsto 50K$, $loc \mapsto$ Seattle,

2. Map $\varphi(\vec{x})^h$ to I/J. Add it to h.

Computing Routes

s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)



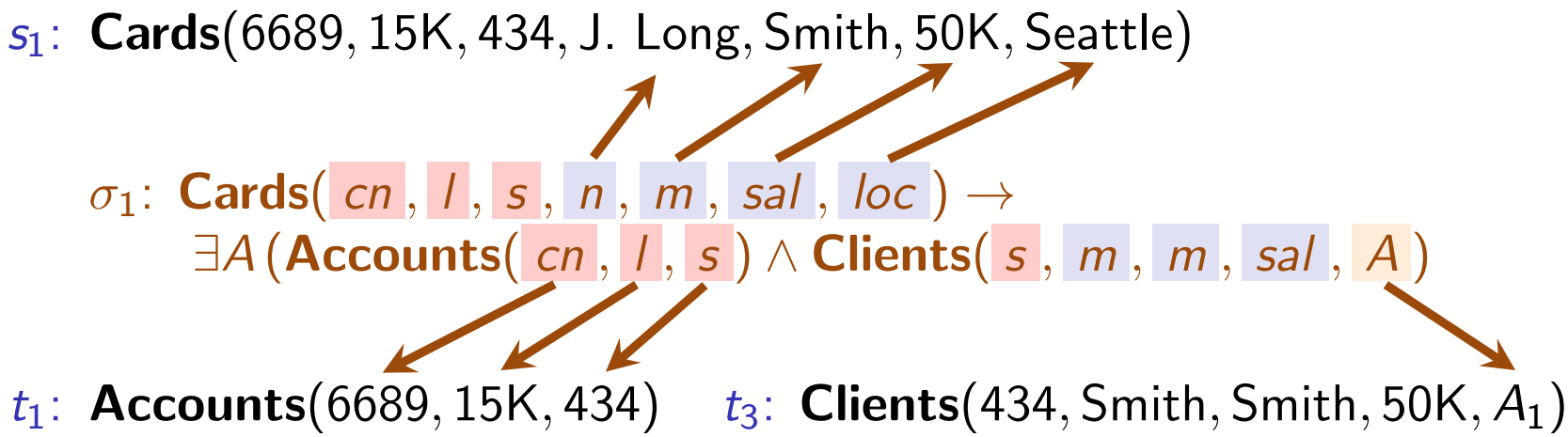
σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434,$
$n \mapsto$ J.Long, $m \mapsto$ Smith, $sal \mapsto 50K$, $loc \mapsto$ Seattle,

2. Map $\varphi(\vec{x})^h$ to I/J. Add it to h.

Computing Routes

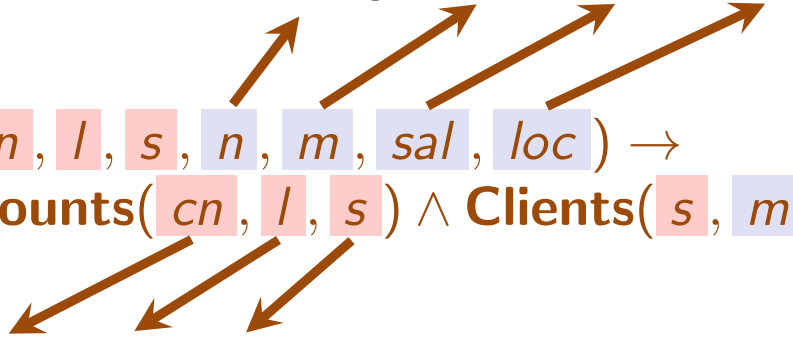


h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434,$
$n \mapsto$ J.Long, $m \mapsto$ Smith, $sal \mapsto 50K$, $loc \mapsto$ Seattle, $A \mapsto A_1\}$

3. Map $\psi(\vec{x}, \vec{y})^h$ to J. Add it to h.

Computing Routes

s_1: **Cards**(6689, 15K, 434, J. Long, Smith, 50K, Seattle)



σ_1: **Cards**(cn , l , s , n , m , sal , loc) $\rightarrow$
$\exists A$ (**Accounts**(cn , l , s) $\wedge$ **Clients**(s , m , m , sal , A)

t_1: **Accounts**(6689, 15K, 434) t_3: **Clients**(434, Smith, Smith, 50K, A_1)

h: $\{cn \mapsto 6689, l \mapsto 15K, s \mapsto 434,$
$n \mapsto$ J.Long, $m \mapsto$ Smith, $sal \mapsto 50K$, $loc \mapsto$ Seattle, $A \mapsto A_1\}$

4 **return** h

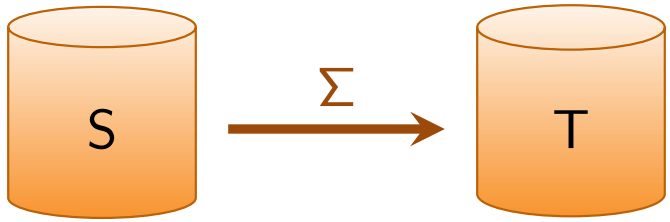
Computing Routes

Algorithm [CT06] (sketch)

FindHom(I, J, t, σ)

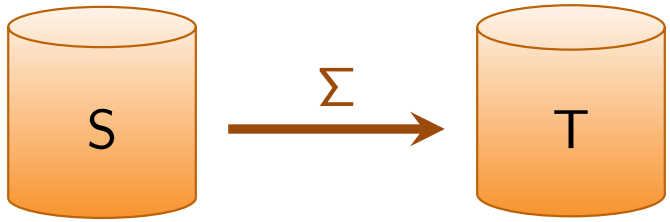
1. Map an atom from $\psi(\vec{x}, \vec{y})$ to t. Add it to h.
2. Map $\varphi(\vec{x})^h$ to I/J. Add it to h.
3. Map $\psi(\vec{x}, \vec{y})^h$ to J. Add it to h.
4. **return** h

Outline



- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
- Optimizing with Relaxed Notions of Equivalence
- Comparing Schema Mappings

Outline



- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
- Optimizing with Relaxed Notions of Equivalence
- Comparing Schema Mappings

Comparing and Optimizing

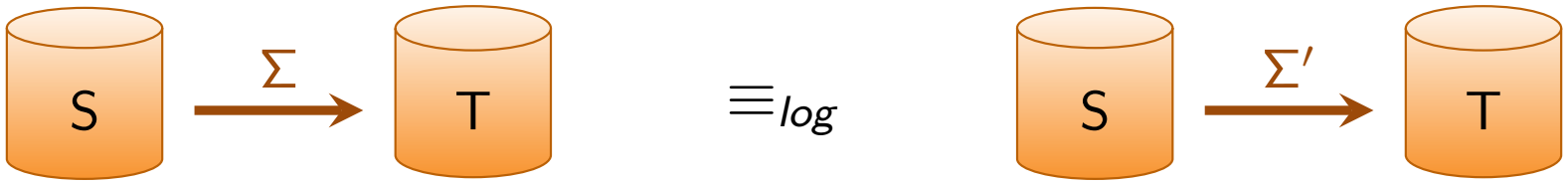
Optimization

Finding a "better" schema mapping that is still "equivalent".

Comparing and Optimizing

Optimization

Finding a "better" schema mapping that is still "equivalent".

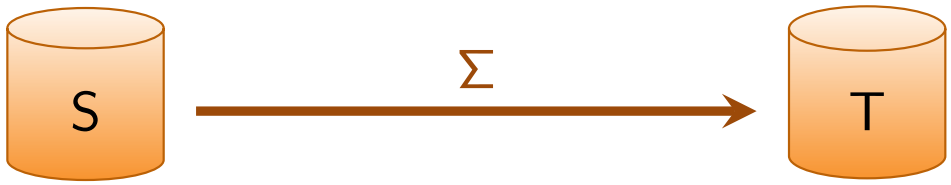


Definition [FKNP08]

$\mathcal{M}$ and $\mathcal{M}'$ are **logically equivalent**, if for every instance (I, J)

$$(I, J) \models \mathcal{M} \quad \Leftrightarrow \quad (I, J) \models \mathcal{M}'$$

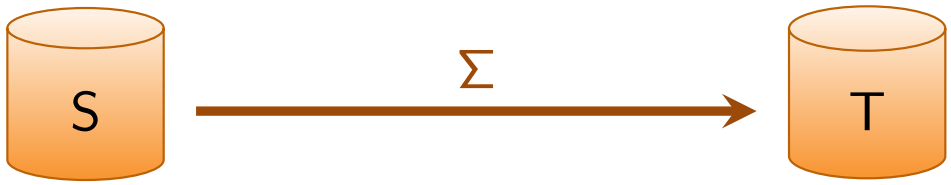
Optimality Criteria



Lecture(*title*, *year*, *prof*)

Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

Optimality Criteria



Lecture(*title*, *year*, *prof*)

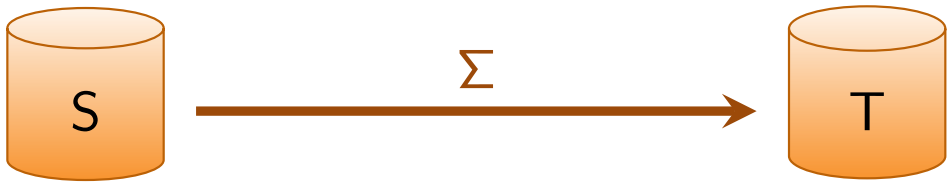
Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

Example

- $L(x_1, x_2, x_3) \rightarrow \exists\, y_1\, , y_2\;\; C(y_1, y_2)\; \wedge C(x_1, y_2)$
- $L(x_1, x_2, x_3) \wedge\; L(x_4, x_5, x_6)\; \rightarrow \exists y_2\; C(x_1, y_2)$

Optimality Criteria

Optimality Criteria



Lecture(*title*, *year*, *prof*)

Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

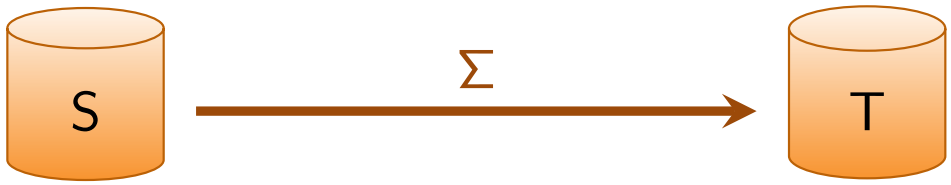
Example

- $L(x_1, x_2, x_3) \rightarrow \exists\, y_1\, , y_2\ \cancel{C(y_1, y_2)} \wedge C(x_1, y_2)$
- $L(x_1, x_2, x_3) \wedge\ L(x_4, x_5, x_6)\ \rightarrow \exists y_2\ C(x_1, y_2)$

Optimality Criteria

- Minimize the number of atoms in each **conclusion**

Optimality Criteria



Lecture(*title*, *year*, *prof*)

Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

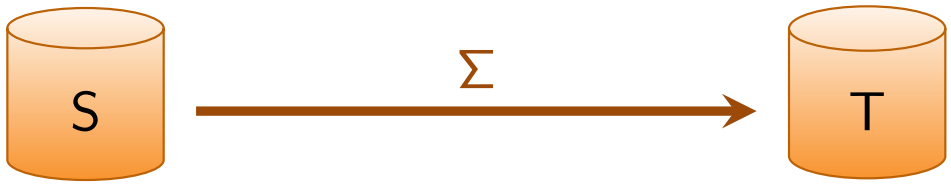
Example

- $L(x_1, x_2, x_3) \rightarrow \exists \cancel{x_1}, y_2 \; \cancel{C(y_1, y_2)} \wedge C(x_1, y_2)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_5, x_6) \rightarrow \exists y_2 \; C(x_1, y_2)$

Optimality Criteria

- Minimize the number of atoms in each **conclusion**
- Minimize the number of **existentially quantified variables**

Optimality Criteria



Lecture(*title*, *year*, *prof*)

Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

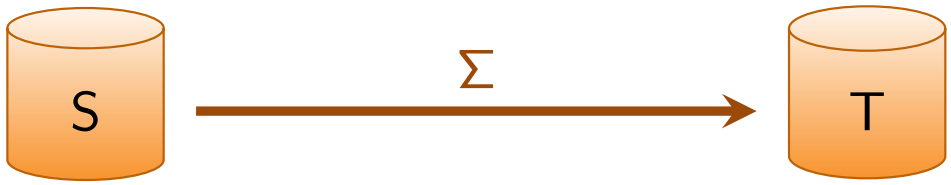
Example

- $L(x_1, x_2, x_3) \rightarrow \exists \cancel{x_1}, y_2 \; \cancel{C(y_1, y_2)} \wedge C(x_1, y_2)$
- $L(x_1, x_2, x_3) \wedge \cancel{L(x_4, x_5, x_6)} \rightarrow \exists y_2 \; C(x_1, y_2)$

Optimality Criteria

- Minimize the number of atoms in each **conclusion**
- Minimize the number of **existentially quantified variables**
- Minimize the number of atoms in each **antecedent**

Optimality Criteria



Lecture(*title*, *year*, *prof*)

Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

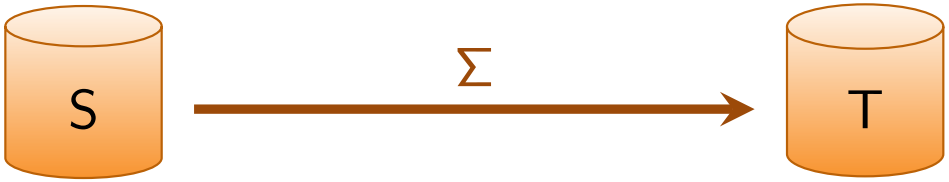
Example

- $L(x_1, x_2, x_3) \rightarrow \exists y_2\ C(x_1, y_2)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_5, x_6) \rightarrow \exists y_2\ C(x_1, y_2)$

Optimality Criteria

- Minimize the number of atoms in each **conclusion**
- Minimize the number of **existentially quantified variables**
- Minimize the number of atoms in each **antecedent**
- Minimize the number of **dependencies**

Optimality Criteria



Lecture(*title*, *year*, *prof*)

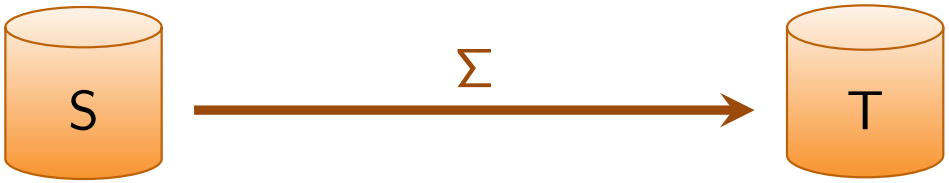
Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

Example

Consider the set Σ of s-t tgds:

- $L(x_1, x_2, x_3) \rightarrow \exists y\, C(x_1, y)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow E(x_1, x_4)$

Optimality Criteria



Lecture(*title*, *year*, *prof*)

Course(*title*, *prof-area*)
Equal-Year(*course*1, *course*2)

Example

Consider the set Σ of s-t tgds:

- $L(x_1, x_2, x_3) \rightarrow \exists y\, C(x_1, y)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow E(x_1, x_4)$

Equivalent set of s-t tgds Σ':

- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow \exists y\ C(x_1, y) \wedge E(x_1, x_4)$

Example (continued)

Consider the set Σ of s-t tgds:

- $L(x_1, x_2, x_3) \rightarrow \exists y\, C(x_1, y)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow E(x_1, x_4)$

Equivalent set of s-t tgds Σ':

- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow \exists y\ C(x_1, y) \wedge E(x_1, x_4)$

Example (continued)

Consider the set Σ of s-t tgds:

- $L(x_1, x_2, x_3) \rightarrow \exists y\, C(x_1, y)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow E(x_1, x_4)$

Equivalent set of s-t tgds Σ':

- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow \exists y\ C(x_1, y) \wedge E(x_1, x_4)$

Observation

Canonical universal solution:

- for Σ: one tuple in the C-relation per tuple in the L-relation
- for Σ': in total, quadratically many tuples in the C-relation

Example (continued)

Consider the set Σ of s-t tgds:

- $L(x_1, x_2, x_3) \rightarrow \exists y\, C(x_1, y)$
- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow E(x_1, x_4)$

Equivalent set of s-t tgds Σ':

- $L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow \exists y\ C(x_1, y) \wedge E(x_1, x_4)$

Observation

Canonical universal solution:

- for Σ: one tuple in the C-relation per tuple in the L-relation
- for Σ': in total, quadratically many tuples in the C-relation

Optimality Criteria

- **Splitting** should be applied whenever possible.

Optimality Criteria

Optimality Criteria

Splitting: Splitting should be applied whenever possible.

Optimality Criteria

Optimality Criteria

Splitting: Splitting should be applied whenever possible.

Optimization goals:

- cardinality-minimality:
  the number of dependencies shall be minimal
- antecedent-minimality:
  the total size of the antecedents shall be minimal
- conclusion-minimality:
  the total size of the conclusions shall be minimal
- variable-minimality:
  the total number of existentially quantified variables shall be minimal

Optimizing Schema Mappings

Rewrite System for s-t tgds [GPS09]

1. Simplification of the conclusion (core computation)
2. Simplification of the antecedent (core computation)
3. Splitting
4. Deletion of an s-t tgd (implication test)
5. Simplification of the conclusion using other tgds (implication test)

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3) \wedge R(y_1, x_2, y_2)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2) \wedge R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3) \wedge R(y_1, x_2, y_2)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2) \wedge R(x_2, y_4, x_3)$

1. Simplification of the conclusion (core computation)

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2) \wedge R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2) \wedge R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2) \wedge R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2) \wedge R(x_2, y_4, x_3)$

3 Splitting

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$

3 Splitting

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$

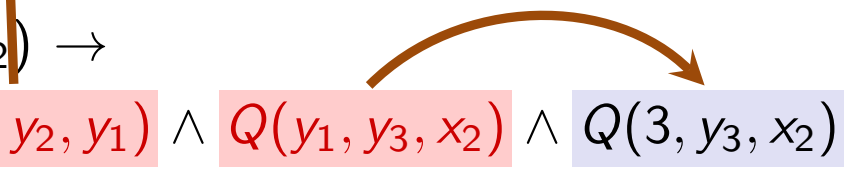
2. Simplification of the antecedent (core computation)

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \rightarrow$
  $P(x_1, y_2, y_1) \wedge Q(y_1, y_3, x_2) \wedge Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$



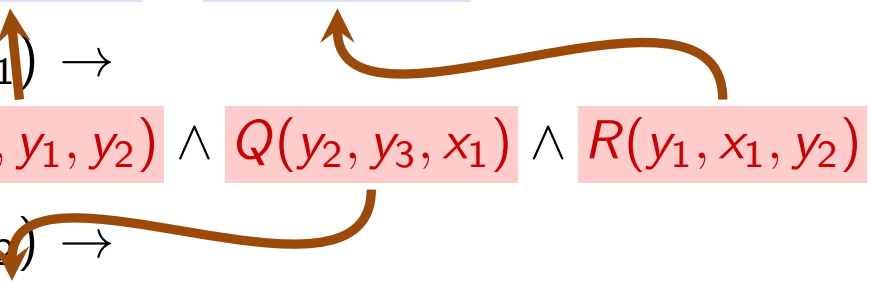
5 Simplification of the conclusion using other tgds (implication test)

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \rightarrow$
  $Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_1, x_1) \rightarrow$
  $P(x_1, y_1, y_2) \wedge Q(y_2, y_3, x_1) \wedge R(y_1, x_1, y_2)$
- $L(x_1, x_2, x_2) \rightarrow$
  $Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$



4 Deletion of an s-t tgd (implication test)

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow$
  $P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$

- $L(x_1, x_2, x_2) \rightarrow$
  $Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow$
  $R(x_2, y_4, x_3)$

Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_2, x_2) \rightarrow Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow R(x_2, y_4, x_3)$

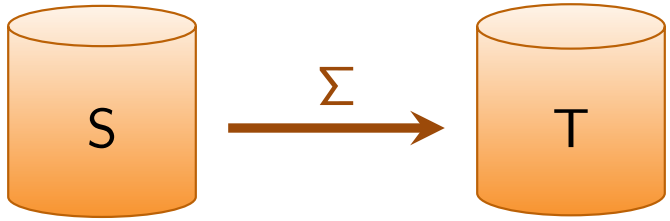
Optimizing Schema Mappings

- $L(x_1, x_2, x_3) \rightarrow P(x_1, y_1, 3) \wedge R(y_1, x_2, 3)$
- $L(x_1, x_2, x_2) \rightarrow Q(3, y_3, x_2)$
- $L(x_1, x_2, x_2) \wedge L(x_1, x_2, x_3) \rightarrow R(x_2, y_4, x_3)$

Result of rewrite system

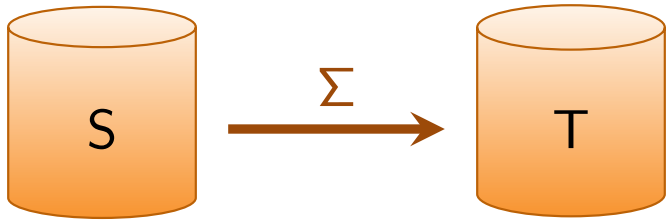
- Is among all logically equivalent split-reduced mappings
  cardinality/antecedent/conclusion/variable-minimal
- Is a unique normal form

Outline



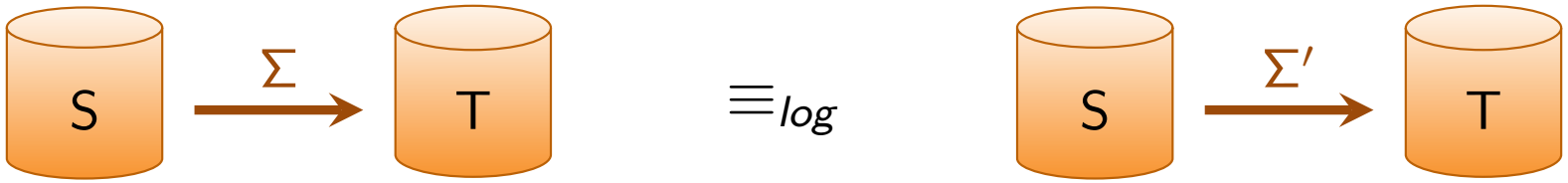
- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
  - Optimality Criteria
  - Optimization and Normalization
- Optimizing with Relaxed Notions of Equivalence
- Comparing Schema Mappings

Outline



- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
  - Optimality Criteria
  - Optimization and Normalization
- Optimizing with Relaxed Notions of Equivalence
- Comparing Schema Mappings

Equivalence

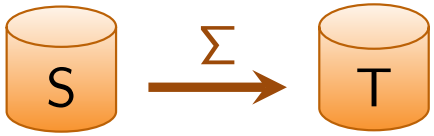


Definition [FKNP08]

$\mathcal{M}$ and $\mathcal{M}'$ are **logically equivalent**, if for every source instance I

$$\mathrm{Sol}(I, \mathcal{M}) = \mathrm{Sol}(I, \mathcal{M}')$$

Equivalence



$S(x) \rightarrow T(x)$

$T'(x, y) \rightarrow T'(y, x)$

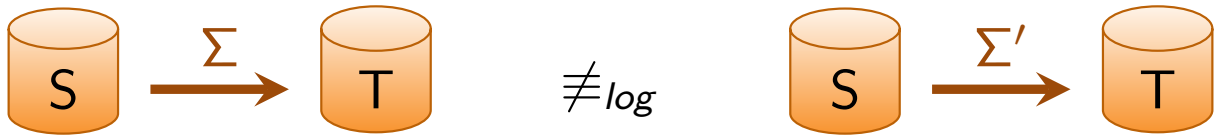
$S(x) \rightarrow T(x)$

Equivalence

$S(x) \rightarrow T(x)$
$T'(x, y) \rightarrow T'(y, x)$

$S(x) \rightarrow T(x)$

Equivalence



$S(x) \to T(x)$
$T'(x,y) \to T'(y,x)$

$S(x) \to T(x)$

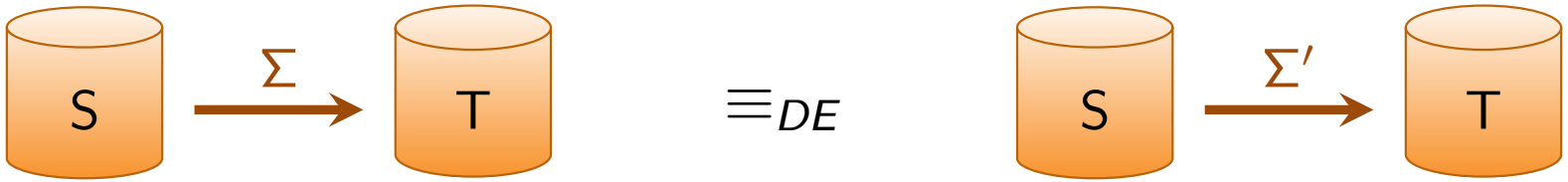
Observation

If we are interested in typical data exchange, i.e. the universal solutions, $\mathcal{M}'$ is "just as good as" $\mathcal{M}$, and has smaller cardinality.

Relaxed Notions of Equivalence

DE equivalence

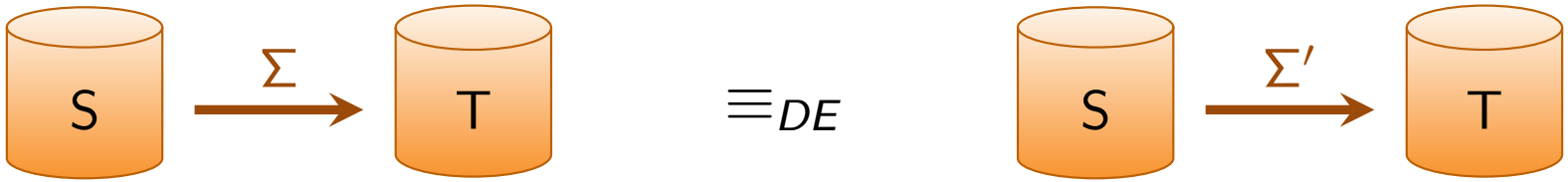
Data-exchange (DE) equivalence does not distinguish mappings which behave in the same way for data exchange.



Relaxed Notions of Equivalence

DE equivalence

Data-exchange (DE) equivalence does not distinguish mappings which behave in the same way for data exchange.



Definition [FKNP08]

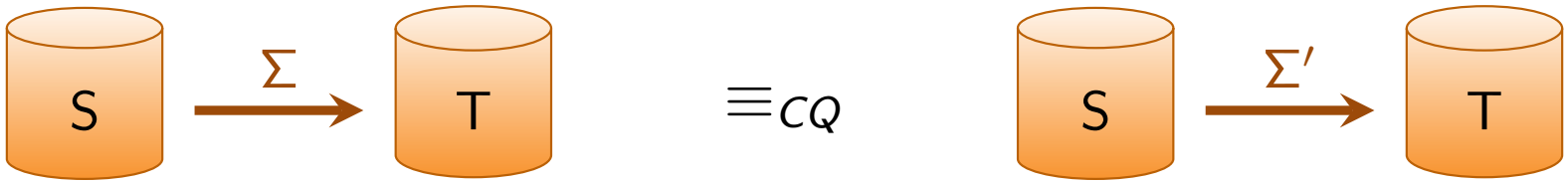
$\mathcal{M}$ and $\mathcal{M}'$ are **data-exchange equivalent**, if for every source instance I

$$\text{UnivSol}(I, \mathcal{M}) = \text{UnivSol}(I, \mathcal{M}')$$

Relaxed Notions of Equivalence

CQ equivalence

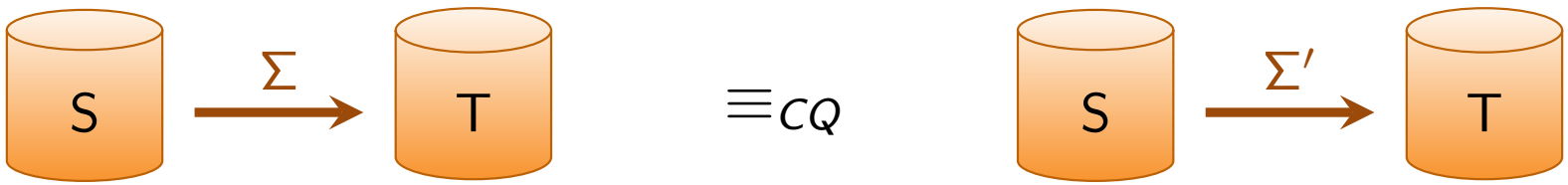
Conjunctive-query (CQ) equivalence does not distinguish mappings which behave similarly for answering conjunctive queries.



Relaxed Notions of Equivalence

CQ equivalence

Conjunctive-query (CQ) equivalence does not distinguish mappings which behave similarly for answering conjunctive queries.

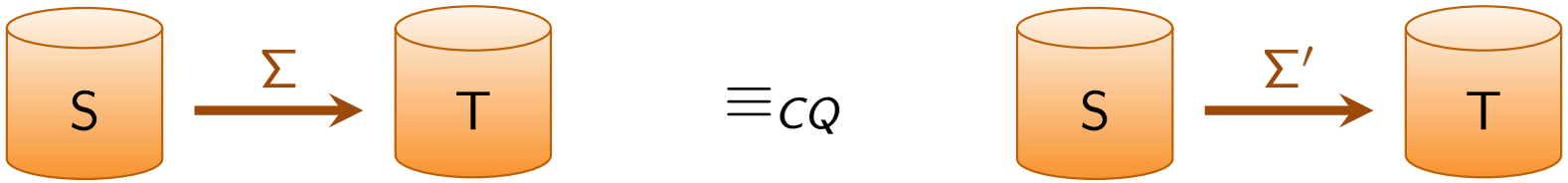


Definition [FKNP08]

$\mathcal{M}$ and $\mathcal{M}'$ are **conjunctive-query equivalent**, if for every source instance I and every CQ q, either $\mathrm{Sol}(I, \mathcal{M}) = \mathrm{Sol}(I, \mathcal{M}') = \emptyset$ or

$$\mathrm{cert}(q, I, \mathcal{M}) = \mathrm{cert}(q, I, \mathcal{M}')$$

Relaxed Notions of Equivalence



Proposition [FKNP08]

Assume that the following holds for every source instance I:

$$\mathrm{Sol}(I, \mathcal{M}) \neq \emptyset \;\Rightarrow\; \mathrm{UnivSol}(I, \mathcal{M}) \neq \emptyset$$

Then $\mathcal{M}$ and $\mathcal{M}'$ are **conjunctive-query equivalent**, if for every source instance I, either $\mathrm{Sol}(I, \mathcal{M}) = \mathrm{Sol}(I, \mathcal{M}') = \emptyset$ or

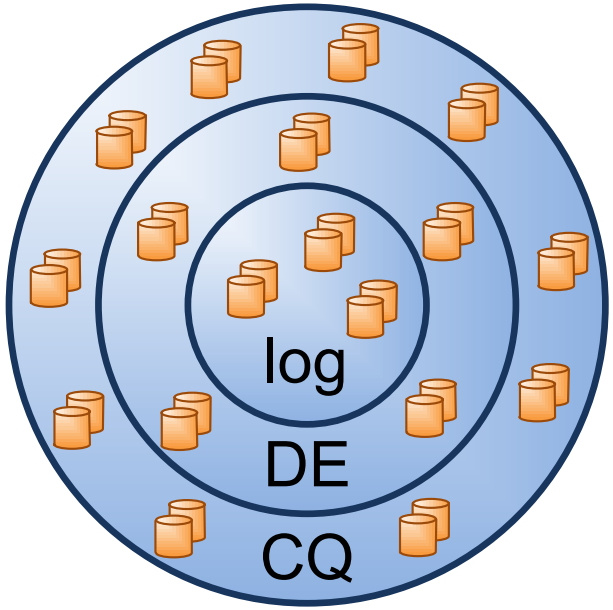
$$\mathrm{core}(I, \mathcal{M}) = \mathrm{core}(I, \mathcal{M}')$$

Hierarchy of Equivalences

Proposition [FKNP08]

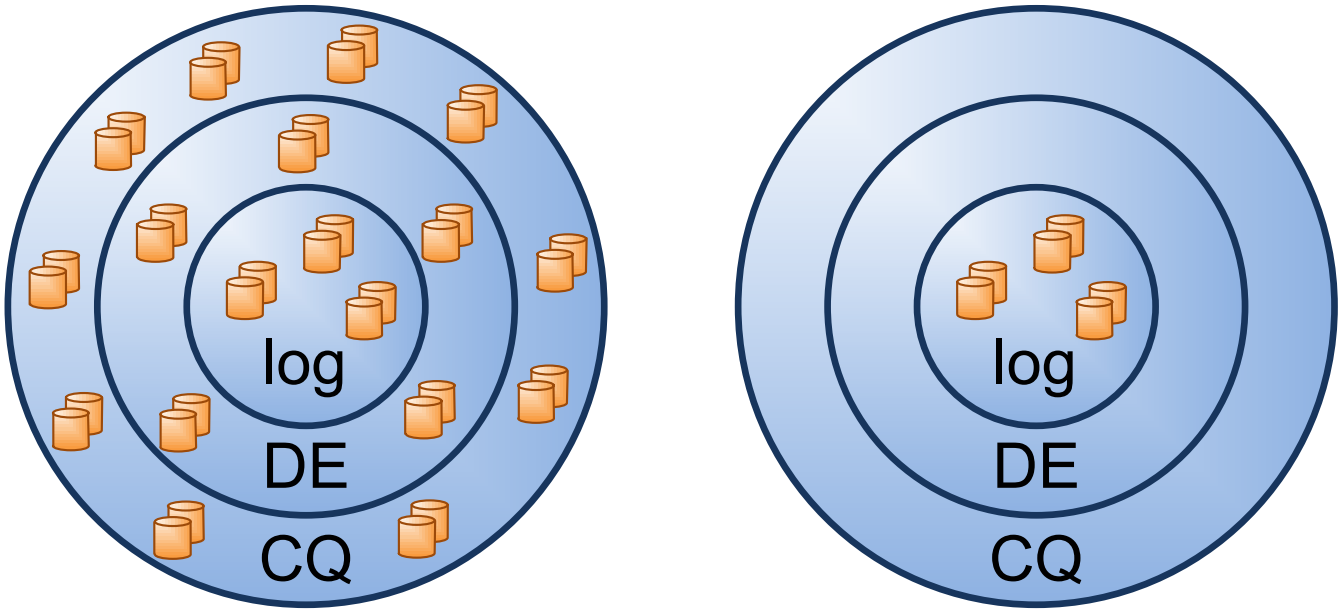
Let $\mathcal{M} = (S, T, \Sigma)$ and $\mathcal{M}' = (S, T, \Sigma')$ be two schema mappings.

$$\mathcal{M} \equiv_{log} \mathcal{M}' \quad \Rightarrow \quad \mathcal{M} \equiv_{DE} \mathcal{M}' \quad \Rightarrow \quad \mathcal{M} \equiv_{CQ} \mathcal{M}'$$



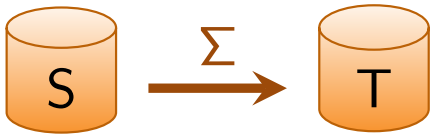
Hierarchy of Equivalences

But is this hierarchy of optimization potential proper, or does it collapse?



This of course depends on the class of schema mappings.

Hierarchy of Equivalences

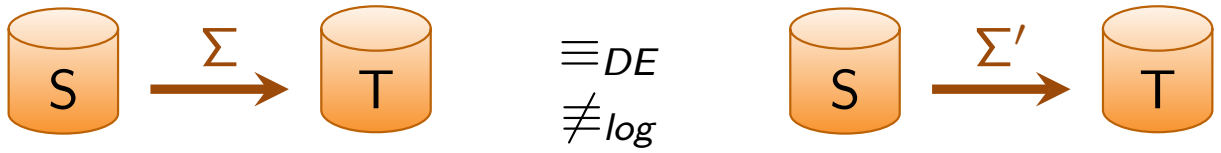


$\emptyset$

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

(s-t tgds and target tgds)

Hierarchy of Equivalences

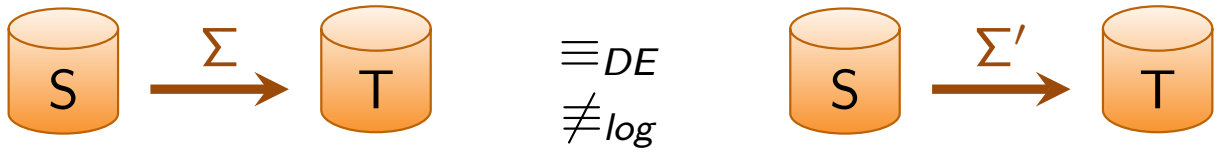


$\emptyset$

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

(s-t tgds and target tgds)

Hierarchy of Equivalences



$\emptyset$

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

(s-t tgds and target tgds)

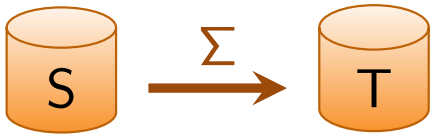
Observation

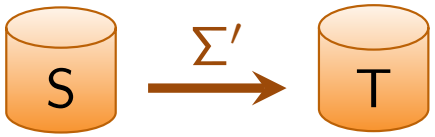
UnivSol$(I, \mathcal{M})$ = UnivSol$(I, \mathcal{M}') = \{\emptyset\}$, however for any I the solution

$$J = \{T(a, b)\}$$

is a solution under $\mathcal{M}$ but not under $\mathcal{M}'$

Hierarchy of Equivalences



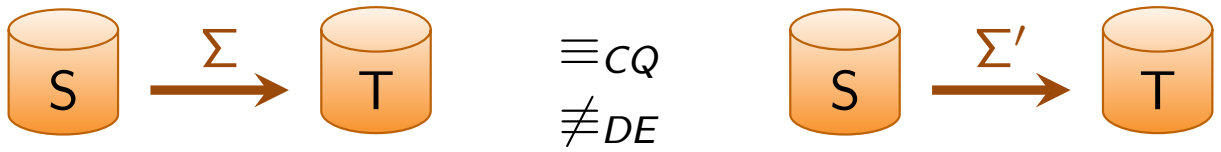


$$S(x) \rightarrow T(x,x)$$
$$T(x,y) \wedge T(y,x) \rightarrow T(x,x)$$

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

(s-t tgds and target tgds)

Hierarchy of Equivalences



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

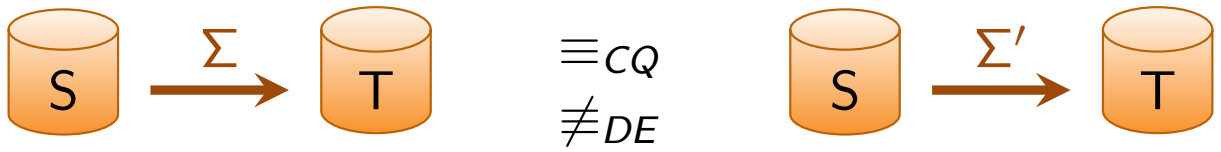
$T(x,y) \wedge T(y,x) \rightarrow T(x,x)$

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

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

(s-t tgds and target tgds)

Hierarchy of Equivalences



$S(x) \rightarrow T(x,x)$
$T(x,y) \wedge T(y,x) \rightarrow T(x,x)$

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

(s-t tgds and target tgds)

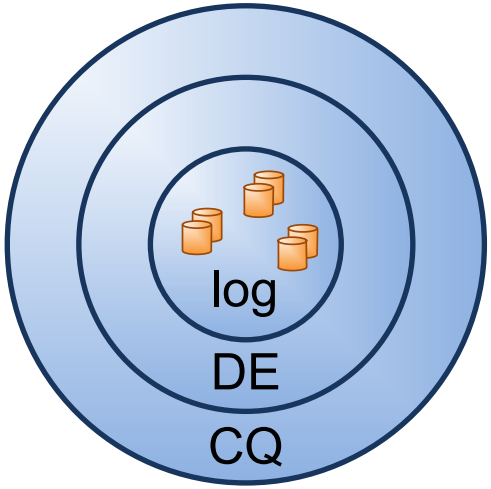
Observation

This is a universal solution for $I = \{S(1)\}$ under $\mathcal{M}$, but not $\mathcal{M}'$:

$$J = \{T(1,1), T(x,y), T(y,z), T(z,x)\}$$

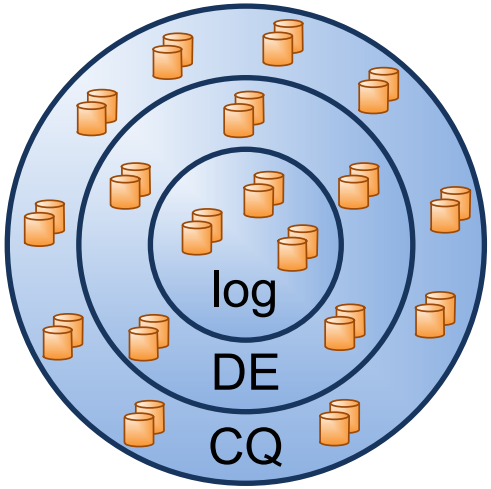
While J is *universal* for $\mathcal{M}$ and $\mathcal{M}'$, J is no solution for I under $\mathcal{M}'$.

Hierarchy of Equivalences



s-t tgds

- no additional optimization power
- all three equivalences decidable



s-t tgds and target tgds

- additional optimization power
- DE- and CQ-equivalence undecidable

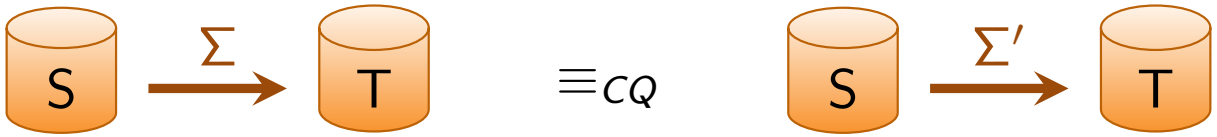
CQ-Equivalence to s-t tgds

Theorem [FKNP08]

If $\mathcal{M}$ is specified by full s-t tgds and full target tgds, then the following statements are equivalent:

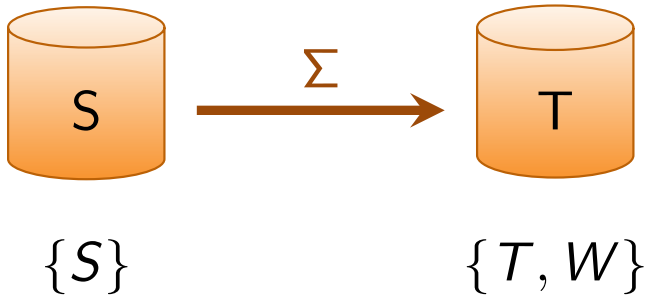
- $\mathcal{M}$ has bounded parallel chase
- There is an $\mathcal{M}' \equiv_{CQ} \mathcal{M}$ specified by full s-t tgds
- There is an $\mathcal{M}' \equiv_{CQ} \mathcal{M}$ specified by s-t tgds
- There is an $\mathcal{M}' \equiv_{CQ} \mathcal{M}$ specified by an SO tgd

CQ-Equivalence to s-t tgds



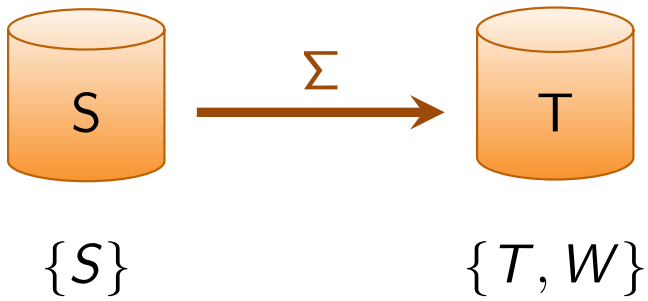
$$
\begin{aligned}
&\exists f \\
&\forall x, y\ S(x,y) \rightarrow T(x,y) \wedge \\
&\forall x \quad S(x,x) \rightarrow T(x,f(x)) \wedge \\
&\forall x \quad S(x,x) \wedge x = f(x) \rightarrow W(x)
\end{aligned}
$$

CQ-Equivalence to s-t tgds



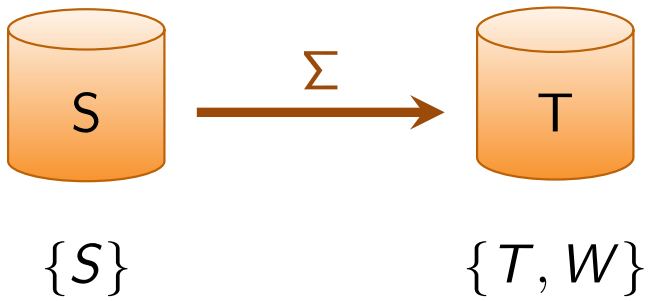
- $S(x, y) \rightarrow T(x, y)$
- $S(x, x) \rightarrow T(x, f(x))$
- $S(x, x) \wedge x = f(x) \rightarrow W(x)$

CQ-Equivalence to s-t tgds



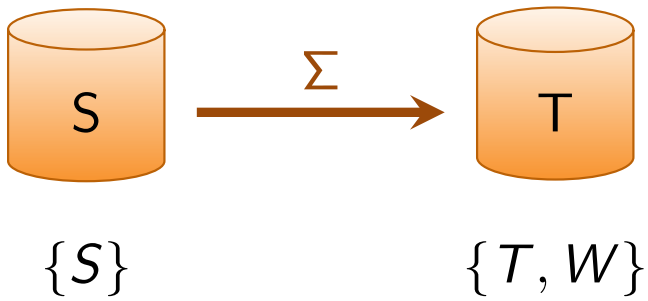
- $S(x, y) \rightarrow T(x, y)$
- $S(x, x) \rightarrow T(x, f(x))$
- $S(x, x) \wedge x = f(x) \rightarrow W(x)$

CQ-Equivalence to s-t tgds



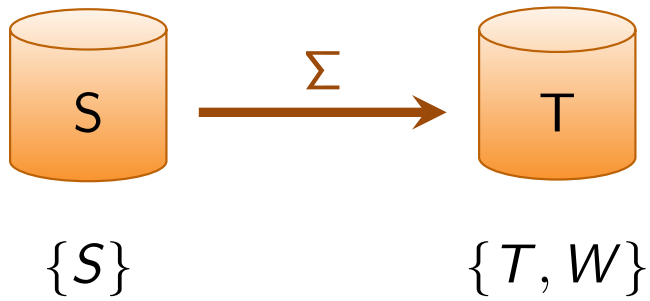
- $S(x, y) \rightarrow T(x, y)$
- $S(x, x) \rightarrow T(x, f(x))$

CQ-Equivalence to s-t tgds



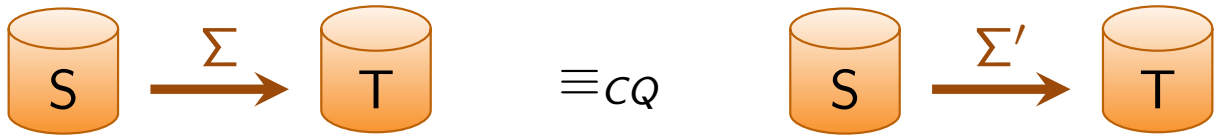
- $S(x, y) \rightarrow T(x, y)$
- $S(x, x) \rightarrow T(x, f(x))$

CQ-Equivalence to s-t tgds



- $S(x, y) \quad \rightarrow T(x, y)$

CQ-Equivalence to s-t tgds



$$\exists f$$
$$\forall x, y\ S(x,y) \rightarrow T(x,y) \wedge$$
$$\forall x\quad S(x,x) \rightarrow T(x, f(x)) \wedge$$
$$\forall x\quad S(x,x) \wedge x = f(x) \rightarrow W(x)$$

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

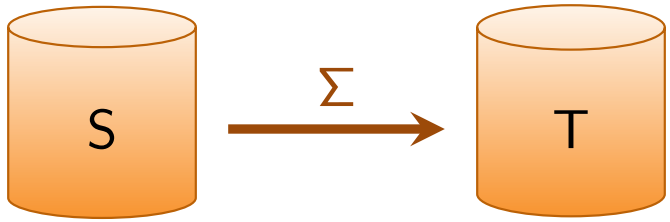
CQ-Equivalence to s-t tgds

Some further results [FKNP08]

Characterization for CQ-equivalence to s-t tgds:

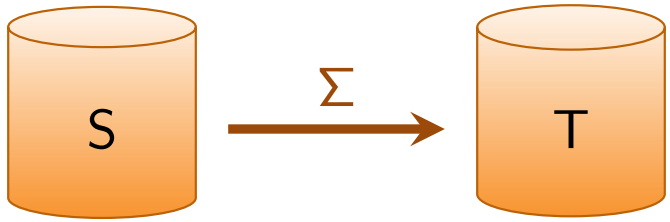
- full s-t tgds and full target tgds: bounded parallel chase
- SO tgds: bounded f-block size
- s-t tgds and target tgds: bounded core chase as well as bounded f-block size

Outline



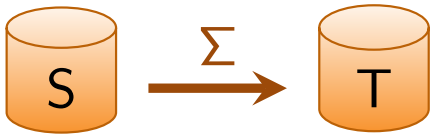
- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
  - Optimality Criteria
  - Optimization and Normalization
- Optimizing with Relaxed Notions of Equivalence
  - Data-Exchange Equivalence
  - Conjunctive-Query Equivalence
- Comparing Schema Mappings

Outline



- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
  - Optimality Criteria
  - Optimization and Normalization
- Optimizing with Relaxed Notions of Equivalence
  - Data-Exchange Equivalence
  - Conjunctive-Query Equivalence
- Comparing Schema Mappings

Information Transfer



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

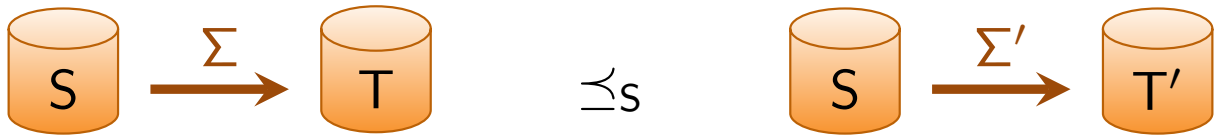
$S(x, y) \rightarrow U(x, y)$

Information Transfer

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

$S(x, y) \rightarrow U(x, y)$

Information Transfer



$S(x, y) \to T(x)$

$S(x, y) \to U(x, y)$

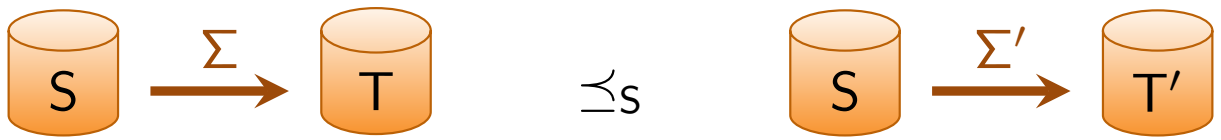
Observation

Intuitively, $\mathcal{M}'$ **transfers more information** than $\mathcal{M}$, since with

- $U(x, y) \to T(x)$

the information transferred by $\mathcal{M}$ can be obtained from the target of $\mathcal{M}'$.

Information Transfer

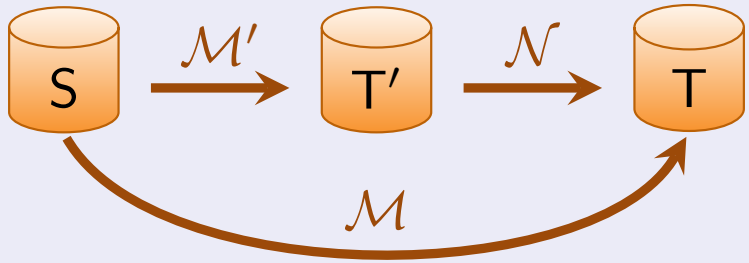


Definition [APRR10]

$\mathcal{M} \preceq_S \mathcal{M}'$ if there exists a mapping $\mathcal{N}$ from T′ to T s.t.

$$\mathcal{M} = \mathcal{M}' \circ \mathcal{N}$$

We say that $\mathcal{M}'$ **transfers as much source information** as $\mathcal{M}$.



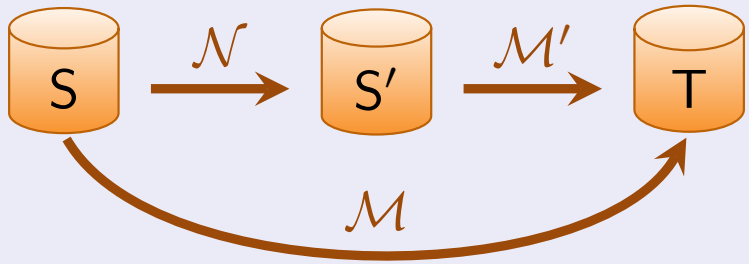
Information Transfer

Definition [APRR10]

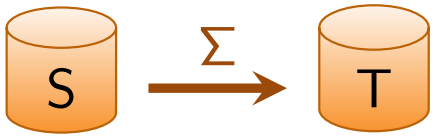
$\mathcal{M} \preceq_{\mathsf{T}} \mathcal{M}'$ if there exists a mapping $\mathcal{N}$ from S to S′ s.t.

$$\mathcal{M} = \mathcal{N} \circ \mathcal{M}'$$

We say that $\mathcal{M}'$ **covers as much target information** as $\mathcal{M}$.



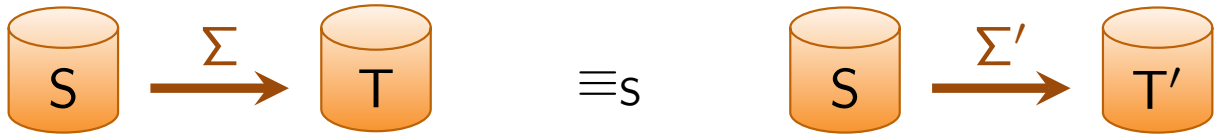
Redundancy



$S(x) \rightarrow T(x)$

$S(x) \rightarrow U(x, x)$

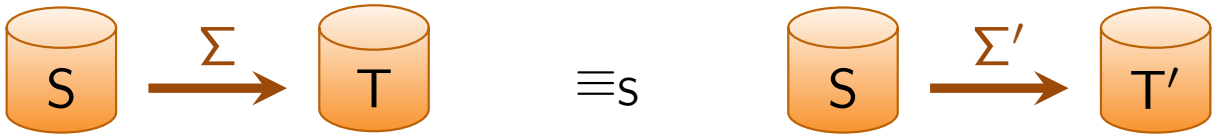
Redundancy



$S(x) \rightarrow T(x)$

$S(x) \rightarrow U(x, x)$

Redundancy



$S(x) \to T(x)$

$S(x) \to U(x, x)$

Definition [APRR10]

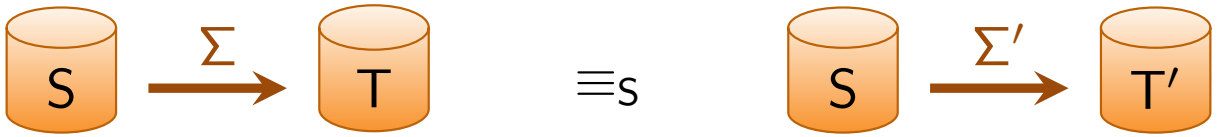
$\mathcal{M}$ is **target redundant** if there exists an instance J^* of T s.t.

$$\mathcal{M}^* = \{(I, J) \in \mathcal{M} \mid J \neq J^*\}$$

satisfies $\mathcal{M}^* \equiv_S \mathcal{M}$.

Similarly defined for **source redundancy**.

Redundancy



$S(x) \to T(x)$

target **non-redundant**

$S(x) \to U(x, x)$

target **redundant**

Definition [APRR10]

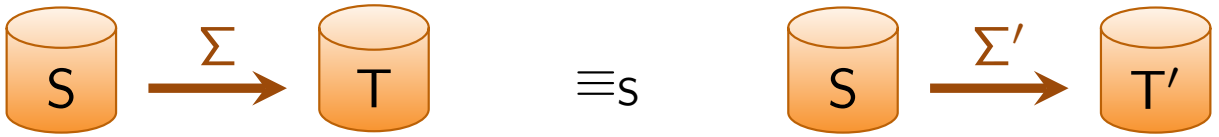
$\mathcal{M}$ is **target redundant** if there exists an instance J^* of T s.t.

$$\mathcal{M}^* = \{(I, J) \in \mathcal{M} \mid J \neq J^*\}$$

satisfies $\mathcal{M}^* \equiv_S \mathcal{M}$.

Similarly defined for **source redundancy**.

Redundancy



$S(x) \rightarrow T(x)$

target **non-redundant**

$S(x) \rightarrow U(x, x)$

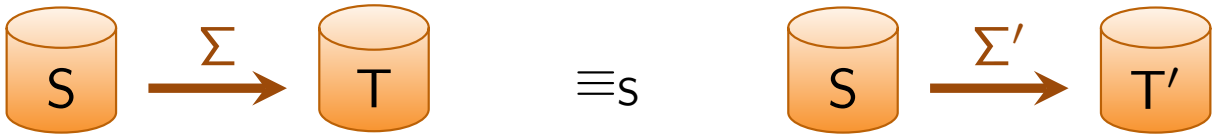
target **redundant**

Observation

Σ' is target redundant, since a solution can contain an atom

$$U(a, b) \text{ with } a \neq b$$

Redundancy



$S(x) \to T(x)$

target **non-redundant**

$S(x) \to U(x, x)$

target **redundant**

Observation

Σ' is target redundant, since a solution can contain an atom

$$U(a, b) \text{ with } a \neq b$$

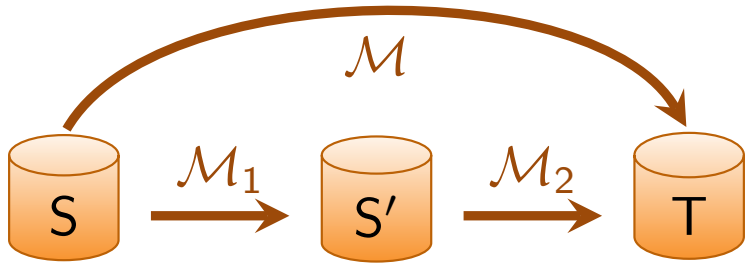
But the schema mapping given as follows is target **non-redundant**:

- $S(x) \to U(x, x)$
- $U(x, y) \to x = y$

Comparing Schema Mappings

Extract operator

Given mapping $\mathcal{M}$, create a new source schema S′ that captures exactly the information participating in $\mathcal{M}$.

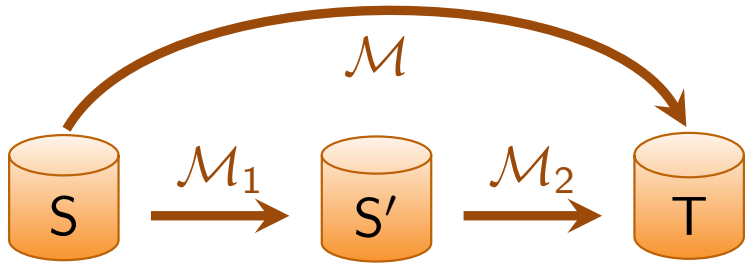


Comparing Schema Mappings

Extract operator

Given mapping $\mathcal{M}$, create a new source schema S′ that captures exactly the information participating in $\mathcal{M}$.

$$P(x, y) \rightarrow \exists u\, T(x, u) \wedge U(x, x)$$
$$P(x, y) \wedge R(y, z) \rightarrow \exists v\, V(x, y, v)$$

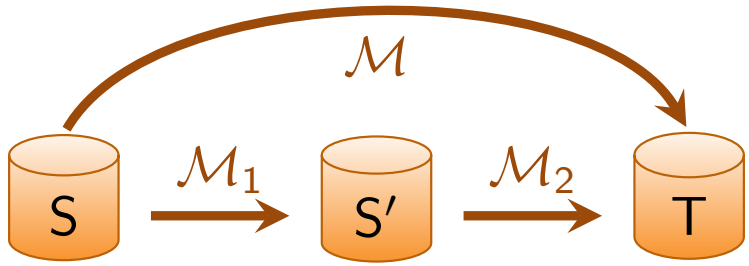


Comparing Schema Mappings

Extract operator

Given mapping $\mathcal{M}$, create a new source schema S′ that captures exactly the information participating in $\mathcal{M}$.

$$P(x, y) \rightarrow \exists u\, T(x, u) \wedge U(x, x)$$
$$P(x, y) \wedge R(y, z) \rightarrow \exists v\, V(x, y, v)$$



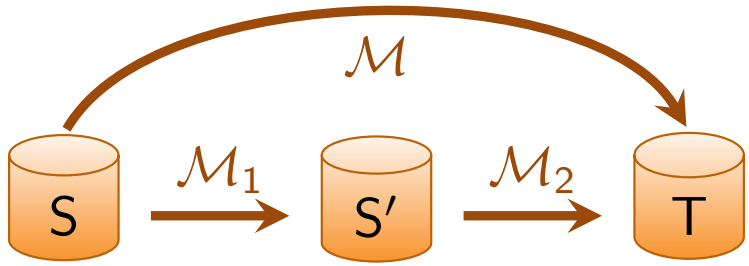
$$P(x, y) \rightarrow P_1(x)$$
$$P(x, y) \wedge R(y, z) \rightarrow P_2(x, y)$$

Comparing Schema Mappings

Extract operator

Given mapping $\mathcal{M}$, create a new source schema S′ that captures exactly the information participating in $\mathcal{M}$.

$$P(x, y) \rightarrow \exists u\, T(x, u) \wedge U(x, x)$$
$$P(x, y) \wedge R(y, z) \rightarrow \exists v\, V(x, y, v)$$



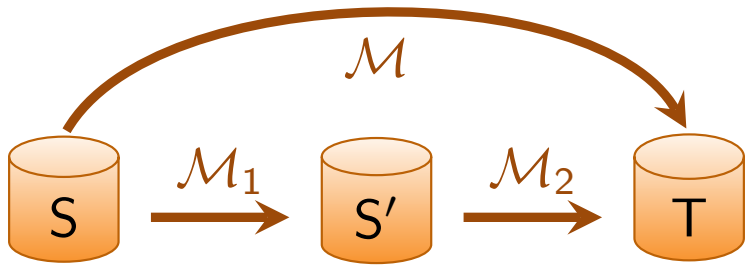
$$P(x, y) \rightarrow P_1(x)$$
$$P(x, y) \wedge R(y, z) \rightarrow P_2(x, y)$$

$$P_1(x) \rightarrow \exists u\, T(x, u) \wedge U(x, x)$$
$$P_2(x, y) \rightarrow \exists v\, V(x, y, v)$$

Comparing Schema Mappings

Extract operator

Given mapping $\mathcal{M}$, create a new source schema S′ that captures exactly the information participating in $\mathcal{M}$.

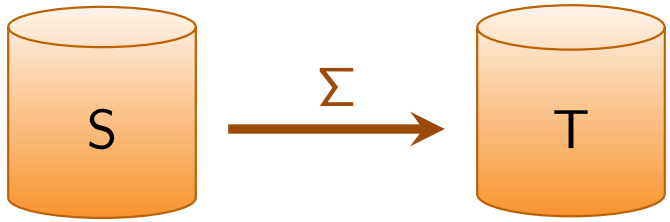


Characterization [APRR10]

$(\mathcal{M}_1, \mathcal{M}_2)$ is an **extract** of $\mathcal{M}$ iff

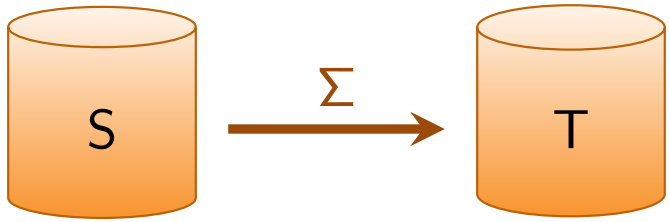
- $\mathcal{M}_1 \circ \mathcal{M}_2 = \mathcal{M}$
- $\mathcal{M}_1 \equiv_{\mathsf{S}} \mathcal{M}$ and $\mathcal{M}_1$ is target non-redundant
- $\mathcal{M}_2 \equiv_{\mathsf{T}} \mathcal{M}$ and $\mathcal{M}_2$ is source non-redundant

Outline



- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
  - Optimality Criteria
  - Optimization and Normalization
- Optimizing with Relaxed Notions of Equivalence
  - Data-Exchange Equivalence
  - Conjunctive-Query Equivalence
- Comparing Schema Mappings
  - Information Transfer
  - Redundancy

Outline



- Analyzing and Debugging
  - Debugging with Routes
  - Computing Routes
- Optimizing with Logical Equivalence
  - Optimality Criteria
  - Optimization and Normalization
- Optimizing with Relaxed Notions of Equivalence
  - Data-Exchange Equivalence
  - Conjunctive-Query Equivalence
- Comparing Schema Mappings
  - Information Transfer
  - Redundancy

Further Results

- Analyzing and Debugging
  - Computing a single route fast
  - Application to XML-based settings
- Optimizing with Logical Equivalence
  - Extending normalization to target egds
- Optimizing with Relaxed Notions of Equivalence
  - Boundary between DE- and CQ-equivalence
  - Full characterization of CQ-equivalence
- Comparing Schema Mappings
  - Characterization of the inverse operator, schema evolution

Next Steps

- Support debugging with routes for target egds
- Extend optimization to target egds
- Normalize and optimize target tgds
- Create a full characterization for DE-equivalence
- Find useful decidable fragments for the relaxed notions
- Develop heuristic approaches to optimization
- Expand results beyond the relational setting

References


Marcelo Arenas, Jorge Pérez, Juan L. Reutter, and Cristian Riveros.
Foundations of schema mapping management.
In Jan Paredaens and Dirk Van Gucht, editors, *PODS*, pages 227–238. ACM, 2010.


Laura Chiticariu and Wang Chiew Tan.
Debugging schema mappings with routes.
In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 79–90. ACM, 2006.


Ronald Fagin, Phokion G. Kolaitis, Alan Nash, and Lucian Popa.
Towards a theory of schema-mapping optimization.
In Maurizio Lenzerini and Domenico Lembo, editors, *PODS*, pages 33–42. ACM, 2008.


Georg Gottlob, Reinhard Pichler, and Vadim Savenkov.
Normalization and optimization of schema mappings.
PVLDB, 2(1):1102–1113, 2009.