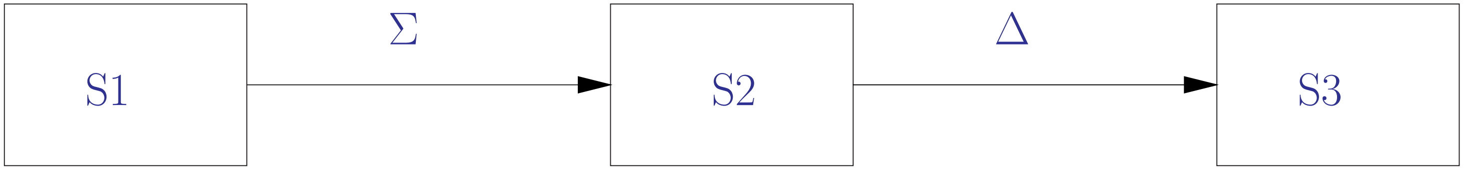
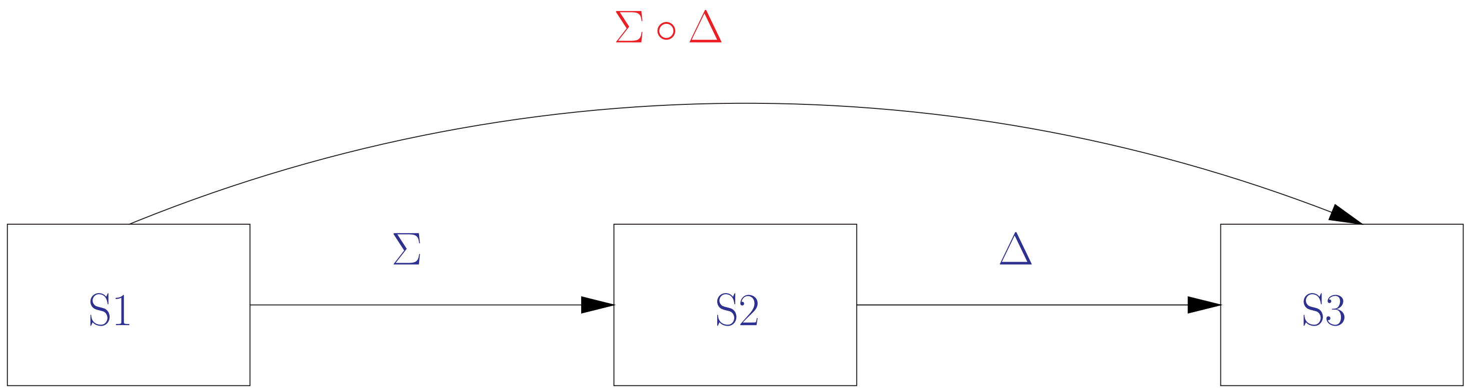
Schema mappings

- Rules used in data exchange specify mappings between schemas.
- To understand the evolution of data one needs to study operations on schema mappings.
- Most commonly we need to deal with two operations:
  - composition
  - inverse

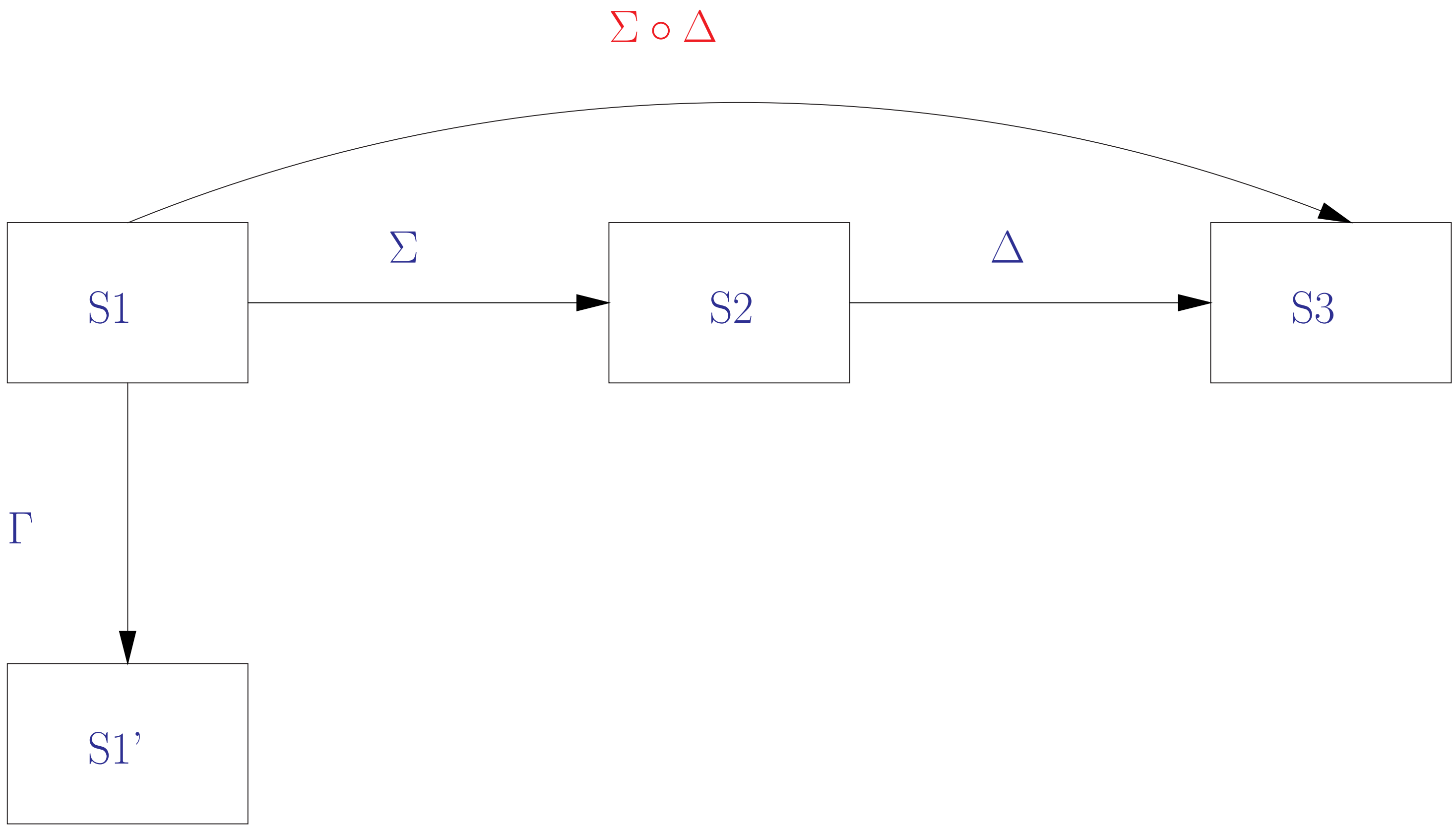
Composition and inverse



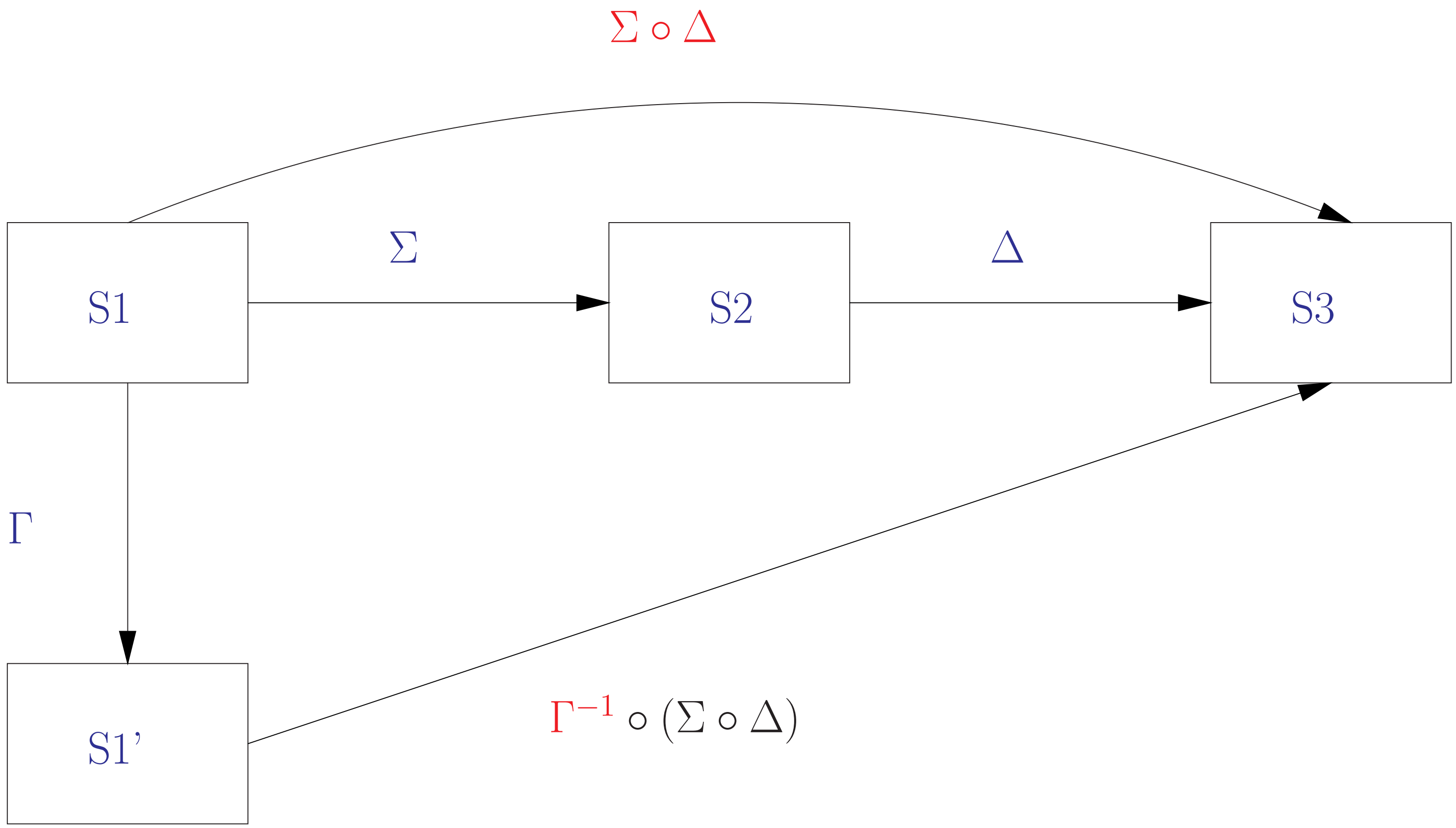
Composition and inverse



Composition and inverse



Composition and inverse



Mappings

- Schema mappings are typically given by rules

$$\psi(\bar{x}, \bar{z}) \ \text{:--}\ \exists \bar{u}\ \varphi(\bar{x}, \bar{y}, \bar{u})$$

  where

  - ψ is a conjunction of atoms over the target:

$$T_1(\bar{x}_1, \bar{z}_1) \wedge \ldots \wedge T_m(\bar{x}_m, \bar{z}_m)$$

  - φ is a conjunction of atoms over the source:

$$S_1(\bar{x}'_1, \bar{y}_1, \bar{u}_1) \wedge \ldots \wedge S_k(\bar{x}'_k, \bar{y}_k, \bar{u}_k)$$

- Example: *Served*(x_1, x_2, z_1, z_2) :– $\exists u_1, u_2$ *Route*$(x_1, u_1, u_2) \wedge$ *BG*(x_1, x_2)

The closure problem

- Are mappings closed under
  - composition?
  - inverse?
- If not, what needs to be added?
- It turns out that mappings are not closed under inverses and composition.
- We next see what might need to be added to them.

Skolem functions

- Source: EP(empl_name,dept,project);
  Target: EDPH(empl_id,dept,phone), DP(dept,project)
- A natural mapping is:

$$\text{EDPH}(z_1, x_2, z_3) \wedge \text{DP}(x_2, x_3) \text{ :– } \text{EP}(x_1, x_2, x_3)$$

- This is problematic: if we have tuples

$$(\text{John, CS, } P_1) \qquad (\text{John, CS, } P_2)$$

  in EP, the canonical solution would have

EDPH

$\bot_1$	CS	$\bot_1'$
$\bot_2$	CS	$\bot_2'$

  corresponding to two projects P_1 and P_2.
- So empl_id is hardly an id!

Skolem functions cont'd

- Solution: make empl_id a function of empl_name.
- Such "invented" functions are called Skolem functions (see Logic 001 for a proper definition)
- Source: EP(empl_name,dept,project);
  Target: EDPH(empl_id,dept,phone), DP(dept,project)
- A new mapping is:

$$\text{EDPH}(f(x_1), x_2, z_3) \wedge \text{DP}(x_2, x_3) \text{ :– } \text{EP}(x_1, x_2, x_3)$$

- f assigns a unique id to every name.

Other possible additions

- One can look at more general queries used in mappings.
- Most generally, relational algebra queries, but to be more modest, one can start with just adding inequalities.
- One may also disjunctions: for example, if we want to invert

$$\begin{array}{rcl} T(x) & :- & S_1(x) \\ T(x) & :- & S_2(x) \end{array}$$

it seems natural to introduce a rule

$$S_1(x) \vee S_2(x) :- T(x)$$

Composition: definition

- Recall the definition of composition of binary relations R and R':

$$(x, z) \in R \circ R' \quad \Leftrightarrow \quad \exists y : \ (x, y) \in R \text{ and } (y, z) \in R'$$

- A schema mapping Σ for two schemas σ and τ is viewed as a binary relation

$$\mathbf{\Sigma} \quad = \quad \left\{ (S, T) \;\middle|\; \begin{array}{l} S \text{ is a } \sigma\text{-instance} \\ T \text{ is a } \tau\text{-instance} \\ T \text{ is a solution for } S \end{array} \right\}$$

- The composition of mappings Σ from σ to τ and Δ from τ to ω is now

$$\mathbf{\Sigma} \circ \mathbf{\Delta}$$

- Question (closure): is there a mapping Γ between σ and ω so that

$$\mathbf{\Gamma} = \mathbf{\Sigma} \circ \mathbf{\Delta}$$

Composition: when it works

- If Σ
  - does not generate any nulls, and
  - no variables $\bar{u}$ for source formulas
- Example:

$$\begin{array}{lrl} \Sigma: & T(x_1,x_2) \wedge T(x_2,x_3) & \text{:-}\ S(x_1,x_2,x_3) \\ \Delta: & W(x_1,x_2,z) & \text{:-}\ T(x_1,x_2) \end{array}$$

- First modify into:

$$\begin{array}{lrl} \Sigma: & T(x_1,x_2) & \text{:-}\ S(x_1,x_2,x_3) \\ \Sigma: & T(x_2,x_3) & \text{:-}\ S(x_1,x_2,x_3) \\ \Delta: & W(x_1,x_2,z) & \text{:-}\ T(x_1,x_2) \end{array}$$

- Then substitute in the definition of W:

Composition: when it cont'd

$$\begin{array}{lcl} W(x_1,x_2,z) & \text{:-} & S(x_1,x_2,y) \\ W(x_1,x_2,z) & \text{:-} & S(y,x_1,x_2) \end{array}$$

to get $\mathbf{\Sigma} \circ \mathbf{\Delta}$.

Explaining the second rule:

$$\begin{array}{ll} & W(x_1,x_2,z) \\ \longrightarrow & T(x_1,x_2) \qquad \text{using } T(var_1,var_2) \text{ :- } S(var_3,var_1,var_2) \\ \longrightarrow & S(y,x_1,x_2) \end{array}$$

Composition: when it doesn't work

- Schema σ: Takes(st_name, course)
- Schema τ: Takes'(st_name, course), NameId(st_name, st_id)
- Schema ω: Enroll(st_id, course)
- Mapping Σ from σ to τ:

$$\begin{aligned} \mathsf{Takes}'(s,c) &\ \text{:–}\ \mathsf{Takes}(s,c) \\ \mathsf{NameId}(s,i) &\ \text{:–}\ \exists c\ \mathsf{Takes}(s,c) \end{aligned}$$

- Mapping Δ from τ to ω:

$$\mathsf{Enroll}(i,c)\ \text{:–}\ \mathsf{NameId}(s,i) \wedge \mathsf{Takes}'(s,c)$$

- A first attempt at the composition: $\mathsf{Enroll}(i,c)$:– $\mathsf{Takes}(s,c)$

Composition: when it doesn't work cont'd

- What's wrong with Γ: $\quad$ Enroll(i, c) :– Takes(s, c)?
- Student id i depends on both name and course!

Takes:

John	CS1
John	CS2

$\stackrel{\Sigma}{\Rightarrow}$

Takes':

John	CS1
John	CS2

NameId:

John	$\perp$

$\stackrel{\Delta}{\Rightarrow}$ Enroll:

$\perp$	CS1
$\perp$	CS2

But:

Takes:

John	CS1
John	CS2

$\stackrel{\Gamma}{\Rightarrow}$ Enroll:

$\perp_1$	CS1
$\perp_2$	CS2

Composition: when it doesn't work cont'd

- Solution: Skolem functions.
- Γ': $\quad \text{Enroll}(f(s), c) \text{ :– } \text{Takes}(s, c)$
- Then:

$$\text{Takes: } \begin{array}{|c|c|}\hline \text{John} & \text{CS1} \\ \hline \text{John} & \text{CS2} \\ \hline \end{array} \quad \stackrel{\Gamma}{\Rightarrow} \quad \text{Enroll: } \begin{array}{|c|c|}\hline \perp & \text{CS1} \\ \hline \perp & \text{CS2} \\ \hline \end{array}$$

- where $\perp = f(\text{John})$

Composition: another example

- Schema σ: Empl(eid)
- Schema τ: Mngr(eid,mngid)
- Schema ω: Mngr'(eid,mngid), SelfMng(id)
- Mapping Σ from σ to τ:

$$\text{Mngr}(e, m) \ \text{:--}\ \text{Empl}(e)$$

- Mapping Δ from τ to ω:

$$\begin{aligned} \text{Mngr'}(e, m) &\ \text{:--}\ \text{Mngr}(e, m) \\ \text{SelfMng}(e) &\ \text{:--}\ \text{Mngr}(e, e) \end{aligned}$$

- Composition:

$$\begin{aligned} \text{Mngr'}(e, f(e)) &\ \text{:--}\ \text{Empl}(e) \\ \text{SelfMng}(e) &\ \text{:--}\ \text{Empl}(e) \wedge e = f(e) \end{aligned}$$

Composition and Skolem functions

- Schema mappings with Skolem functions compose!
- Algorithm:
  - replace all nulls by Skolem functions
    - Mngr$(e, f(e))$:– Empl(e)
    - Δ stays as before
  - Use substitution:
    - Mngr'(e, m) :– Mngr(e, m) becomes
      Mngr'$(e, f(e))$:– Empl(e)
    - SelfMng(e) :– Mngr(e, e) becomes
      SelfMng(e) :– Empl$(e) \wedge e = f(e)$

Inverting mappings

- Harder than composition.
- Intuition: $\Sigma \circ \Sigma^{-1} = \mathbf{ID}$.
- But even what $\mathbf{ID}$ should be is not entirely clear.
- Some intuitive examples will follow.

Examples of inversion

- The inverse of projection is null invention:
  - $T(x) \text{ :- } S(x,y)$
  - $S(x,y) \text{ :- } T(x)$
- Inverse of union requires disjunction:
  - $T(x) \text{ :- } S(x) \quad T(x) \text{ :- } S'(x)$
  - $S(x) \vee S'(x) \text{ :- } T(x)$
- So reversing the rules doesn't always work.

Examples of inversion cont'd

- Inverse of decomposition is join:
  - $T(x_1, x_2) \wedge T'(x_2, x_3)$:– $S(x_1, x_2, x_3)$
  - $S(x_1, x_2, x_3)$:– $T(x_1, x_2) \wedge T'(x_2, x_3)$
- But this is also an inverse of $T(x_1, x_2) \wedge T'(x_2, x_3)$:– $S(x_1, x_2, x_3)$:
  - $S(x_1, x_2, z)$:– $T(x_1, x_2)$
  - $S(z, x_2, x_3)$:– $T'(x_2, x_3)$

Examples of inversion cont'd

- One may need to distinguish nulls from values in inverses.
- Σ given by

$$\begin{aligned} T_1(x) &\ :\!-\ S(x,x) \\ T_2(x,z) &\ :\!-\ S(x,y) \wedge S(y,x) \\ T_3(x_1,x_2,z) &\ :\!-\ S(x_1,x_2) \end{aligned}$$

- Its inverse Σ^{-1} requires:
  - a predicate NotNull and
  - inequalities:

$$S(x,x)\ :\!-\ T_1(x) \wedge T_2(x,y_1) \wedge T_3(x,y_1,y_2) \wedge \mathsf{NotNull}(x)$$

$$S(x_1,x_2)\ :\!-\ T_3(x_1,x_2,y) \wedge (x_1 \neq x_2) \wedge \mathsf{NotNull}(x_1) \wedge \mathsf{NotNull}(x_2)$$