UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Data Exchange 2

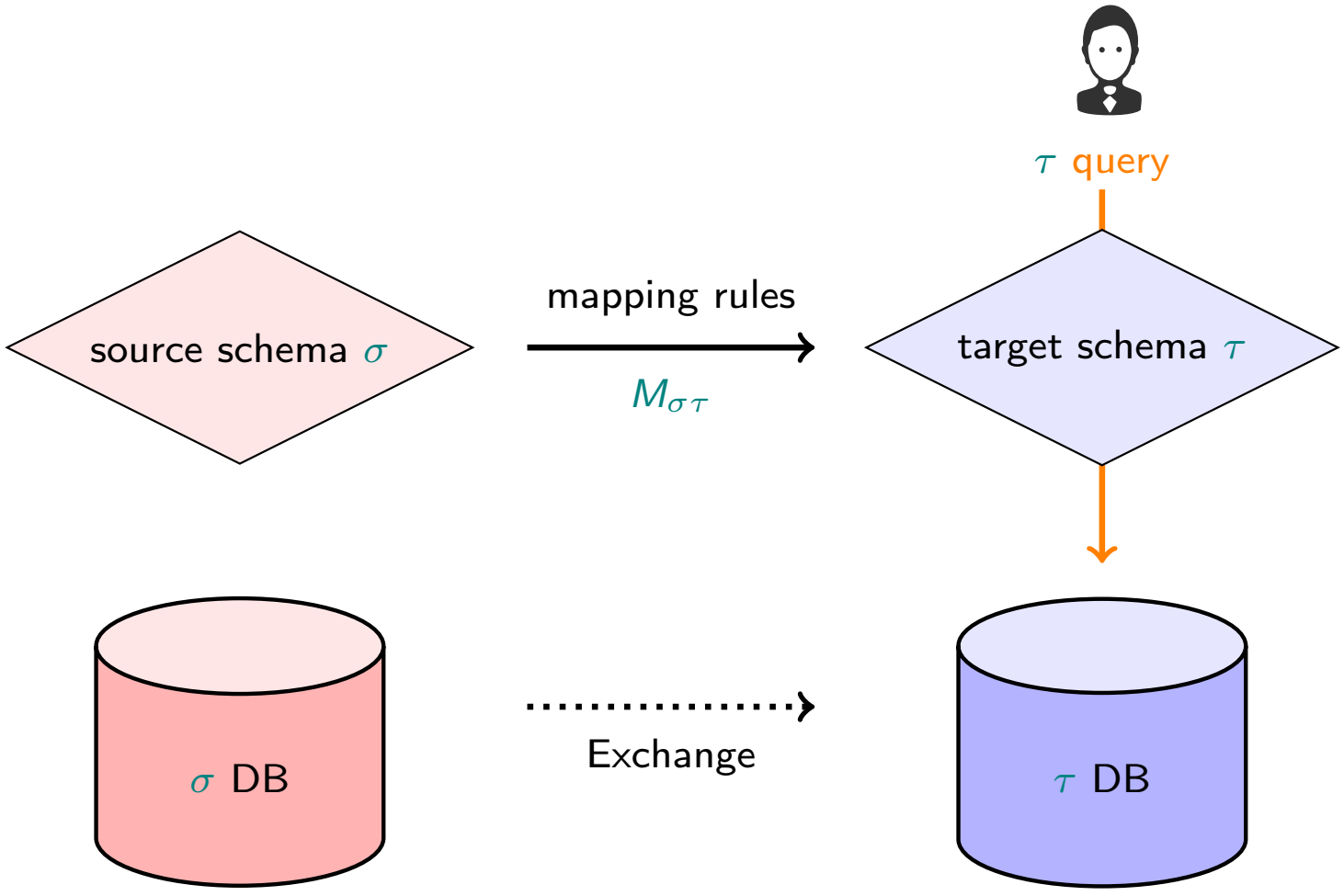
Lecture 6: Universal Solutions, Core, Certain Answers

22 November, 2017

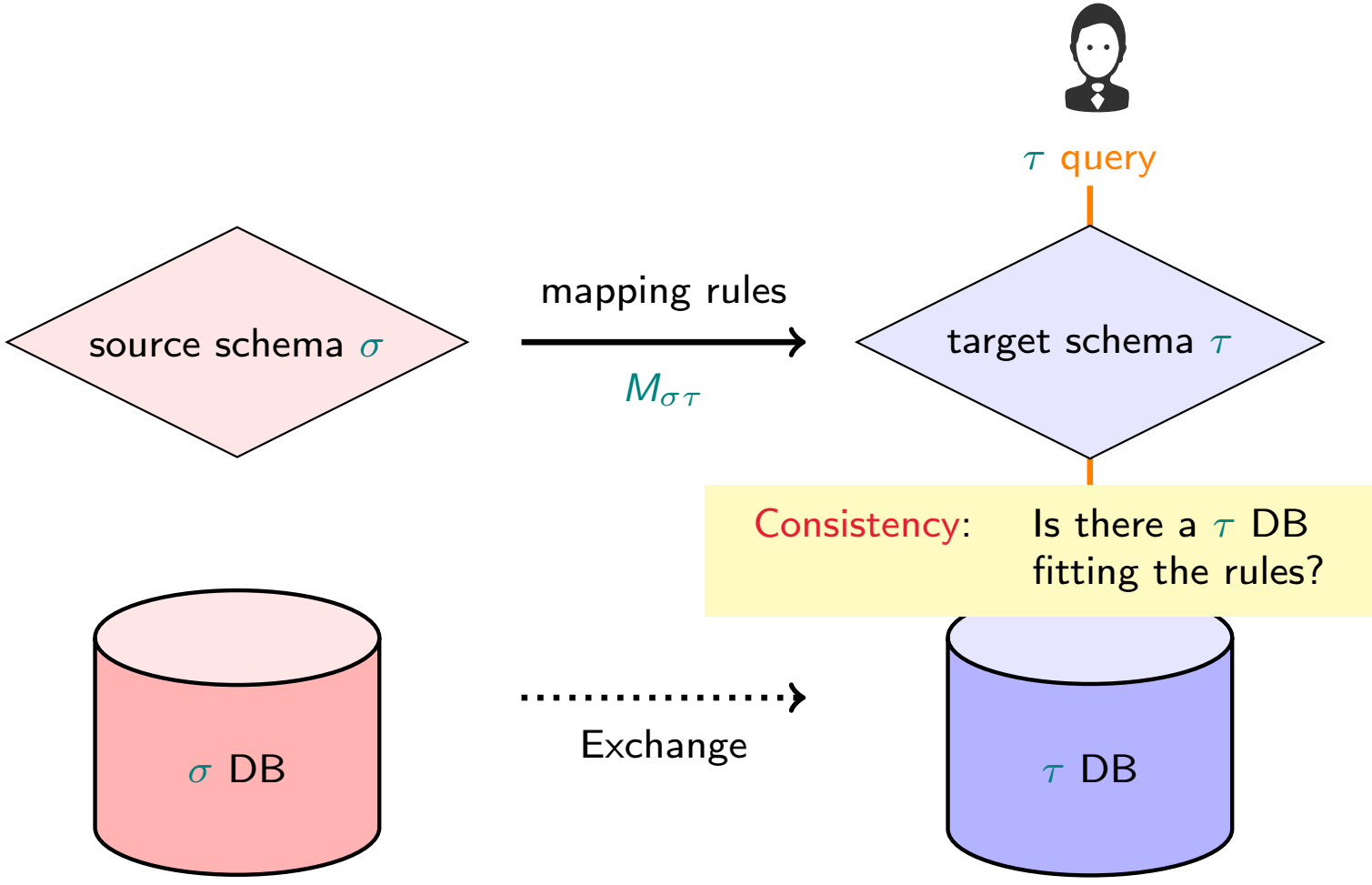
Foundations of Ontologies and Databases for Information Systems

CS5130 (Winter 17/18)

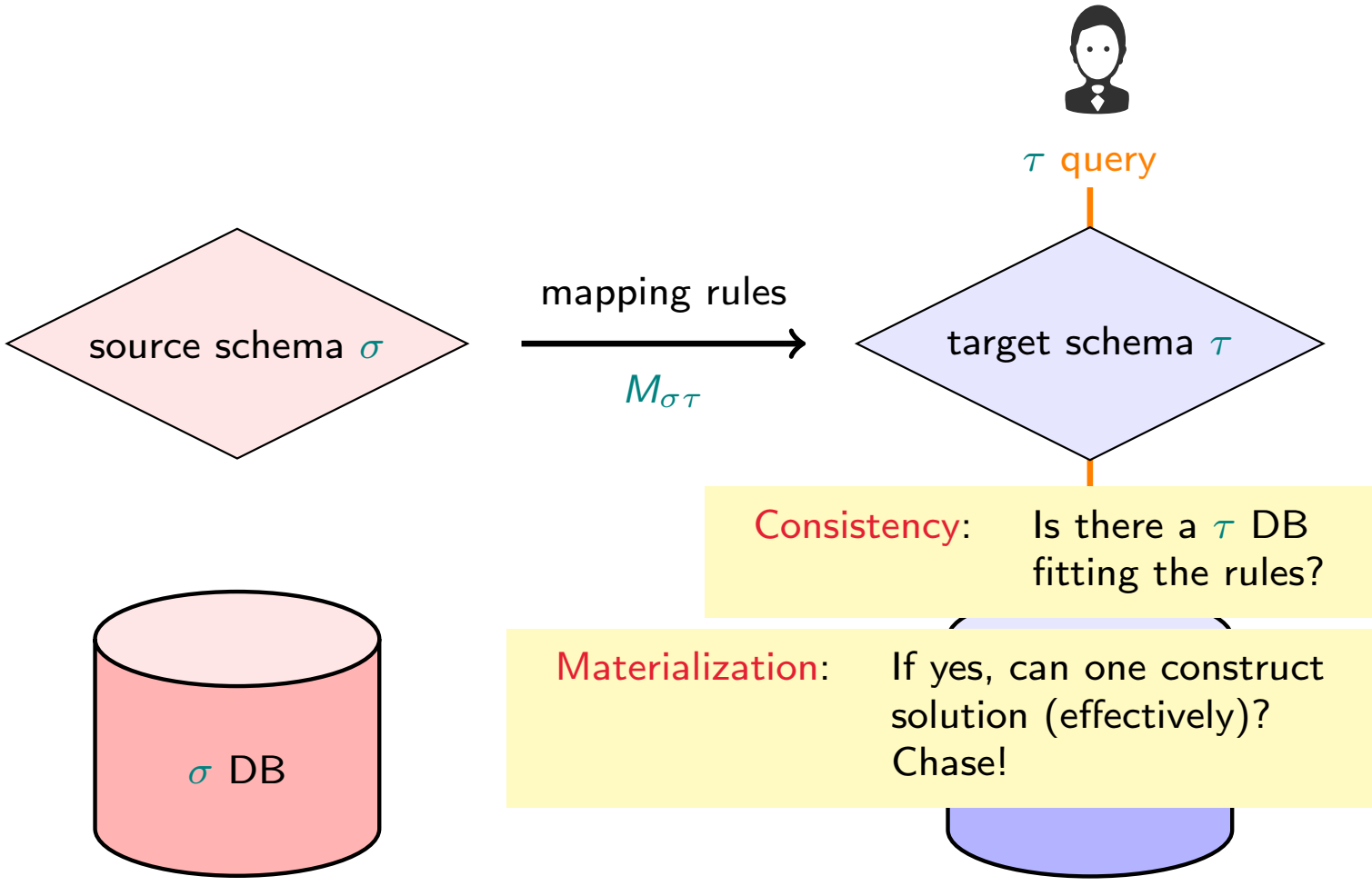
Recap of Lecture 5


τ query
source schema σ
mapping rules
$M_{\sigma\tau}$
target schema τ
σ DB
Exchange
τ DB

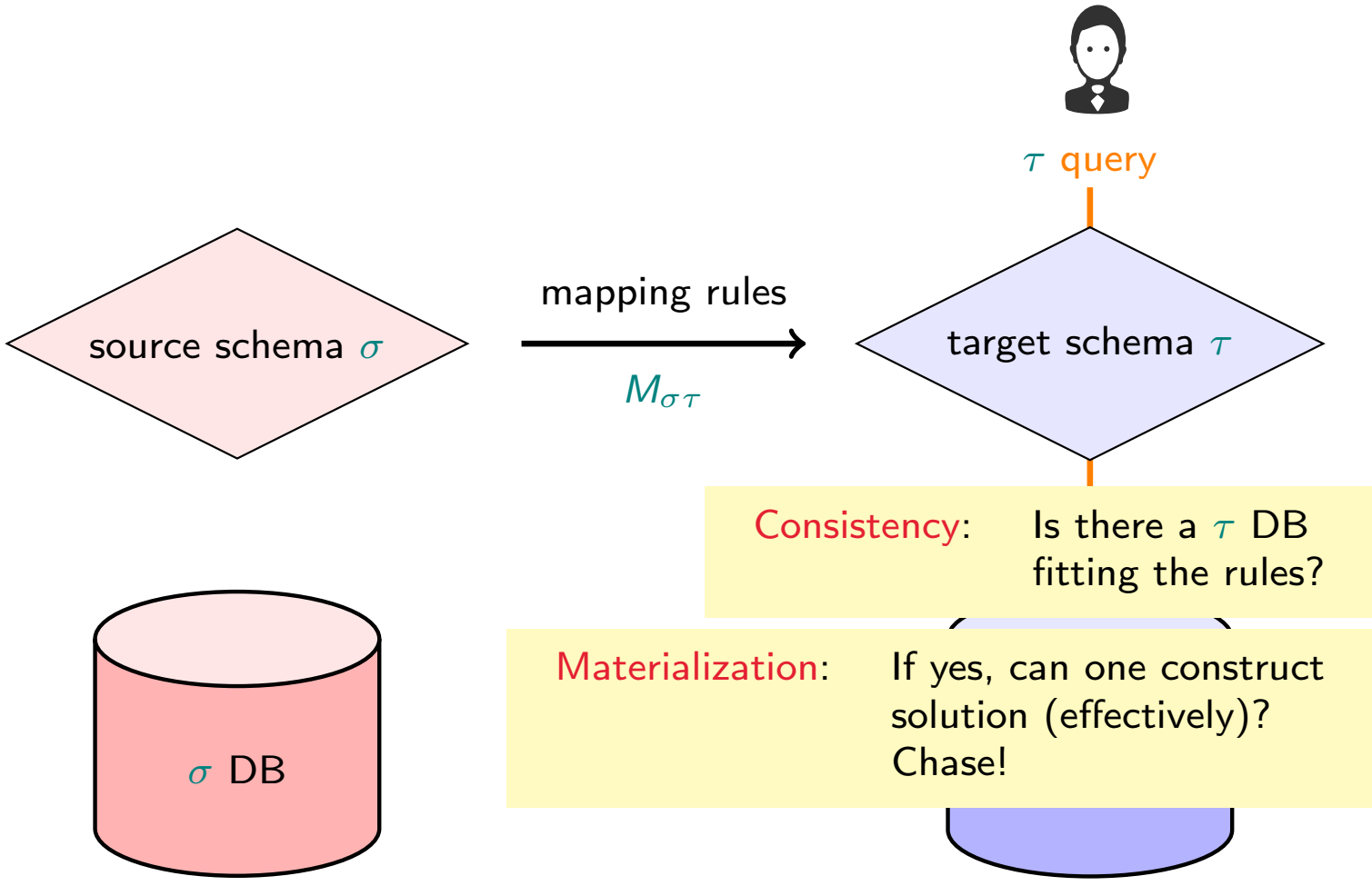
End of Recap


τ query
source schema σ
mapping rules
$M_{\sigma\tau}$
target schema τ
Consistency: Is there a τ DB fitting the rules?
σ DB
Exchange
τ DB

End of Recap


source schema σ
mapping rules
$M_{\sigma\tau}$
target schema τ
τ query
σ DB
Consistency: Is there a τ DB fitting the rules?
Materialization: If yes, can one construct solution (effectively)? Chase!

End of Recap


source schema σ
mapping rules
$M_{\sigma\tau}$
target schema τ
τ query
σ DB
Consistency: Is there a τ DB fitting the rules?
Materialization: If yes, can one construct solution (effectively)? Chase!

End of Recap

Universal Solutions

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city, coun, pop)

Flight(src, dest, airl, dep)

paris	sant.	airFr	2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city, coun, pop)

Flight(src, dest, airl, dep)

paris	sant.	airFr	2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

non-necessary co-reference

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(	city,	coun,	pop)
Flight(	src,	dest,	airl, dep)
	paris	sant.	airFr 2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\}$$

non-necessary instantiation

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city,	coun,	pop)

Flight(src,	dest,	airl,	dep)
paris	sant.	airFr	2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

Why is $\mathfrak{T}$ universal?

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city, coun, pop)

Flight(src, dest, airl, dep)

paris sant. airFr 2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $\mathit{Flight}(\mathit{src}, \mathit{dest}, \mathit{airl}, \mathit{dep}) \longrightarrow \exists \mathit{fno}\, \exists \mathit{arr}\, (\mathit{Route}(\mathit{fno}, \mathit{src}, \mathit{dest}) \wedge \mathit{Info}(\mathit{fno}, \mathit{dep}, \mathit{arr}, \mathit{airl}))$

Universal solutions

$\perp_1 \mapsto \perp_3$

$$\mathfrak{T} = \{\mathit{Route}(\perp_1, \mathit{paris}, \mathit{sant}), \mathit{Info}(\perp_1, 2320, \perp_2, \mathit{airFr})\}$$

$$\mathfrak{T}' = \{\mathit{Route}(\perp_3, \mathit{paris}, \mathit{sant}), \mathit{Info}(\perp_3, 2320, \perp_2, \mathit{airFr})\}$$

$$\mathfrak{T}'' = \{\mathit{Route}(\perp_1, \mathit{paris}, \mathit{sant}), \mathit{Info}(\perp_1, 2320, \perp_1, \mathit{airFr})\}$$

$$\mathfrak{T}''' = \{\mathit{Route}(123, \mathit{paris}, \mathit{sant}), \mathit{Info}(123, 2320, \perp_2, \mathit{airFr})\} \ldots$$

Why is $\mathfrak{T}$ universal?

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city, coun, pop)

Flight(src, dest, airl, dep)

paris	sant.	airFr	2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$$

$\bot_2 \mapsto \bot_1$

$$\mathfrak{T}' = \{Route(\bot_3, paris, sant), Info(\bot_3, 2320, \bot_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \bot_2, airFr)\} \ldots$$

Why is $\mathfrak{T}$ universal?

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city, coun, pop)

Flight(src, dest, airl, dep)

paris	sant.	airFr	2320

Target schema τ

Route(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\bot_3, paris, sant), Info(\bot_3, 2320, \bot_2, airFr)\}$$

$\bot_1 \mapsto 123$

$$\mathfrak{T}'' = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \bot_2, airFr)\} \ldots$$

Why is $\mathfrak{T}$ universal?

Example (DE in Flight Domain)

Source schema σ and instance $\mathfrak{S}$

Geo(city,	coun,	pop)

Flight(src,	dest,	airl,	dep)
paris	sant.	airFr	2320

Target schema τ

Route(fno, src, dest)

Info(fno, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

Any universal solution works: $cert_{\mathcal{M}}(Q, \mathfrak{S}) = Q(\mathfrak{T}) = Q(\mathfrak{T}')$

What are Good Solutions?

- We are seeking universal solutions: they represent all other ones

- A solution $\mathfrak{T}$ may contain NULLs
- A DB instance is complete iff it does not contain NULLs

- $Rep(\mathfrak{T})$ = all complete DBs instances that represent $\mathfrak{T}$
- Explicate "represent" by homomorphism notion
- Now formally define

$$Rep(\mathfrak{T}) = \{\mathfrak{T}' \mid \text{There is } h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}' \text{ for complete } \mathfrak{T}'\}$$

Homomorphism

- Intuitively, homomorphisms are structure preserving mappings
- Defined here for DB instances but similarly definable for arbitrary structures

Definition

A **Homomorphism** $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$ is a map

$$h : Var(\mathfrak{T}) \cup CONST \rightarrow VAR(\mathfrak{T}') \cup CONST$$

s.t.

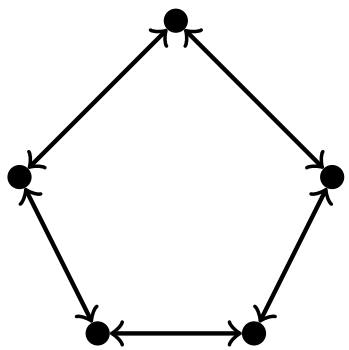
- $h(c) = c$ for all $c \in CONST$ and
- if $R(\vec{t}) \in \mathfrak{T}$, then $R(h(\vec{t})) \in \mathfrak{T}'$

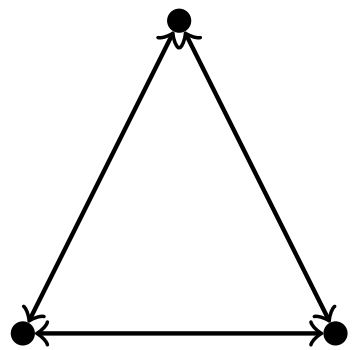
Wake-Up Exercise

Consider two instances that are graphs, namely

- $\mathfrak{G}$ = cycle on 5 nodes with marked nulls $\nu_1, \ldots, \nu_5$
- $\mathfrak{G}'$ = cycle on 3 nodes with marked nulls ν'_1, ν'_2, ν'_3.

Give examples of a mapping $h : \mathfrak{G} \to \mathfrak{G}'$ that is a homomorphism, resp. not a homomorphism.



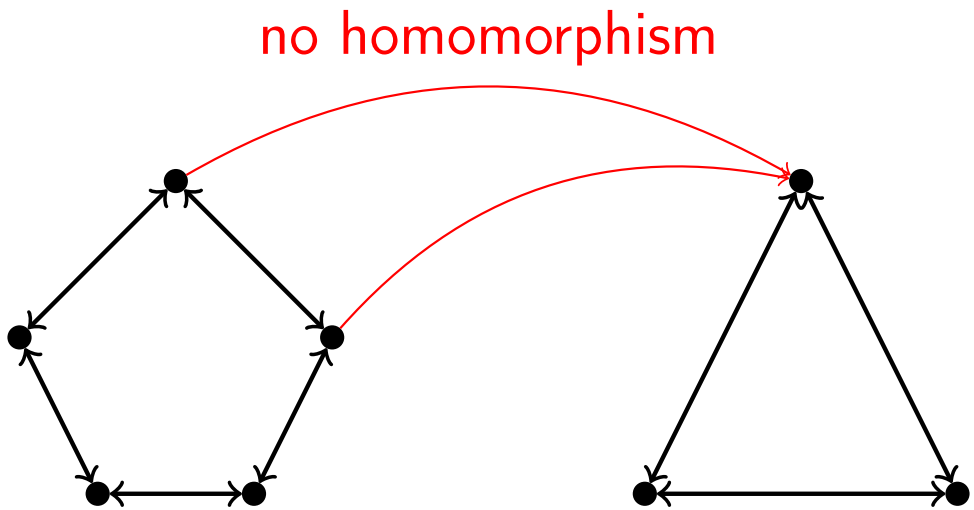


Wake-Up Exercise

Consider two instances that are graphs, namely

- $\mathfrak{G}$ = cycle on 5 nodes with marked nulls $\nu_1, \ldots, \nu_5$
- $\mathfrak{G}'$ = cycle on 3 nodes with marked nulls ν'_1, ν'_2, ν'_3.

Give examples of a mapping $h : \mathfrak{G} \to \mathfrak{G}'$ that is a homomorphism, resp. not a homomorphism.

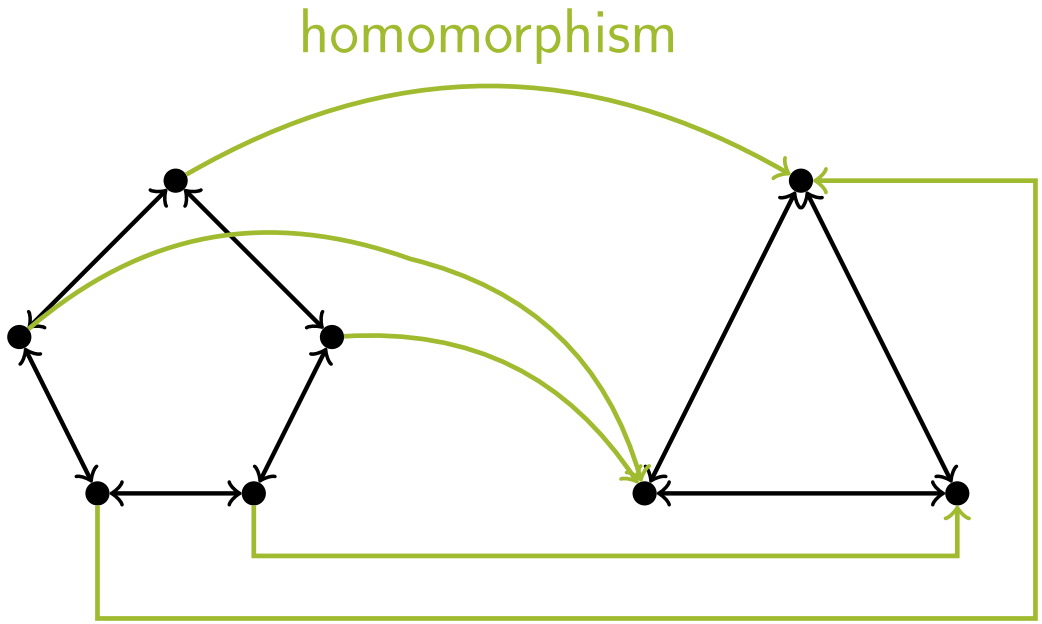


Wake-Up Exercise

Consider two instances that are graphs, namely

- $\mathfrak{G}$ = cycle on 5 nodes with marked nulls $\nu_1, \ldots, \nu_5$
- $\mathfrak{G}'$ = cycle on 3 nodes with marked nulls ν'_1, ν'_2, ν'_3.

Give examples of a mapping $h : \mathfrak{G} \to \mathfrak{G}'$ that is a homomorphism, resp. not a homomorphism.



Universal Solutions

- There are three equivalent characterizations of universal solutions $\mathfrak{T}$; mainly work with third as definition

Definition (Universal Solution)

1. Solution $\mathfrak{T}$ describing all others

$$USol_1(\mathfrak{T}) : \{\mathfrak{T}' \in SOL_{\mathcal{M}}(\mathfrak{S}) \mid \mathfrak{T}' \text{ complete}\} \subseteq Rep(\mathfrak{T})$$

2. Solution $\mathfrak{T}$ as general as all others

$$USol_2(\mathfrak{T}) : Rep(\mathfrak{T}') \subseteq Rep(\mathfrak{T}) \quad \text{for every } \mathfrak{T}' \in SOL_{\mathcal{M}}(\mathfrak{S})$$

3. Solution $\mathfrak{T}$ mapping homomorphically into others

$$USol_3(\mathfrak{T}) : \text{ For all } \mathfrak{T}' \in SOL_{\mathcal{M}}(\mathfrak{S}) \text{ there is } h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$$

Example (Non-existence of Universal Solutions)

- $M_{\sigma\tau} = \{\ E(x,y) \to G(x,y)\ \}$
- $M_\tau = \{\ G(x,y) \to \exists z\ L(y,z), \quad L(x,y) \to \exists z\ G(y,z)\ \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $\mathfrak{T} = \{G(a,b), L(b,a)\}$ is a solution
- But there is no universal solution

Proof sketch (by contradiction)

- A universal solution must have an infinite sequence $(\mathfrak{S}, \{G(a,b), L(b,\nu_1), G(\nu_1,\nu_2), L(\nu_2,\nu_3), G(\nu_3,\nu_4)\ldots\})$
- As $\mathfrak{T}$ is finite there must be some identification of an ν_i with a or b or with another ν_j
- In any case a contradiction follows (by constructing a solution into which no homomorphic embedding of $\mathfrak{T}$ is possible)

Example (Non-existence of Universal Solutions)

- $M_{\sigma\tau} = \{\ E(x,y) \to G(x,y)\ \}$
- $M_{\tau} = \{\ G(x,y) \to \exists z\ L(y,z), \quad L(x,y) \to \exists z\ G(y,z)\ \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $\mathfrak{T} = \{G(a,b), L(b,a)\}$ is a solution
- But there is no universal solution

Proof sketch (by contradiction)

- A universal solution must have an infinite sequence $(\mathfrak{S}, \{G(a,b), L(b,\nu_1), G(\nu_1,\nu_2), L(\nu_2,\nu_3), G(\nu_3,\nu_4)\ldots\})$
- As $\mathfrak{T}$ is finite there must be some identification of an ν_i with a or b or with another ν_j
- In any case a contradiction follows (by constructing a solution into which no homomorphic embedding of $\mathfrak{T}$ is possible)

Example (Non-existence of Universal Solutions)

- $M_{\sigma\tau} = \{\ E(x,y) \to G(x,y)\ \}$
- $M_\tau = \{\ G(x,y) \to \exists z\ L(y,z), \quad L(x,y) \to \exists z\ G(y,z)\ \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $\mathfrak{T} = \{G(a,b), L(b,a)\}$ is a solution
- But there is no universal solution

Proof sketch (by contradiction)

- A universal solution must have an infinite sequence $(\mathfrak{S}, \{G(a,b), L(b,\nu_1), G(\nu_1,\nu_2), L(\nu_2,\nu_3), G(\nu_3,\nu_4)\ldots\})$
- As $\mathfrak{T}$ is finite there must be some identification of an ν_i with a or b or with another ν_j
- In any case a contradiction follows (by constructing a solution into which no homomorphic embedding of $\mathfrak{T}$ is possible)

Example (Non-existence of Universal Solutions)

- $M_{\sigma\tau} = \{\ E(x,y) \to G(x,y)\ \}$
- $M_\tau = \{\ G(x,y) \to \exists z\ L(y,z), \quad L(x,y) \to \exists z\ G(y,z)\ \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $\mathfrak{T} = \{G(a,b), L(b,a)\}$ is a solution
- But there is no universal solution

Proof sketch (by contradiction)

- A universal solution must have an infinite sequence $(\mathfrak{S}, \{G(a,b), L(b,\nu_1), G(\nu_1,\nu_2), L(\nu_2,\nu_3), G(\nu_3,\nu_4)\dots\})$
- Consider case where $\nu_{2i-1} = a$ and define solution $\mathfrak{T}' = \{G(a,b), L(b,c_1), G(c_1,c_2), L(c_2,c_3), \dots, G(c_j,c_{j-1})$ for $2i < j$ and fresh c_i
- There must be an $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$.
- But then $h(\nu_l) = c_l$ and hence $h(\nu_{2i-1}) = c_{2i-1}$, but also $h(\nu_{2i-1}) = h(a) = a$. ↯

Undecidability of Universal Solution Existence

UNISOLEXISTENCE$_{\mathcal{M}}$

- Input: A source instance $\mathfrak{S}$
- Output: Is there a universal solution for $\mathfrak{S}$ under $\mathcal{M}$?

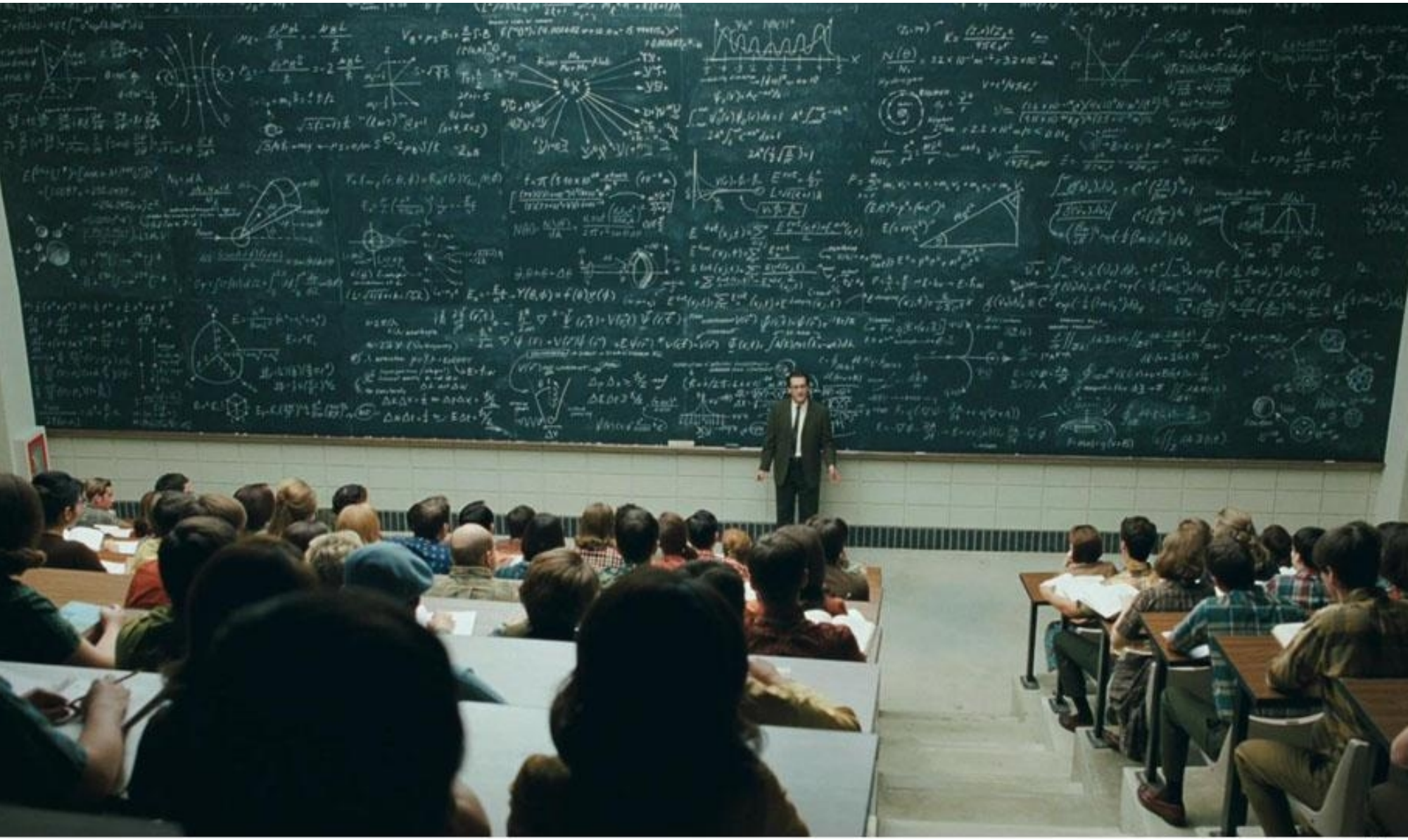
- Allowing arbitrary dependencies leads to undecidability
- Shown by of reduction of halting problem

Theorem

There exists a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ s.t. UNISOLEXISTENCE$_{\mathcal{M}}$ is undecidable

- Proof in book of Arenas et al. 5 pages long, so ... we do not show it here

By the way: There are Longer Proofs



By the Way: There are Longer Proofs

- Recent example: A computer aided proof for a particular case (C = 3) of the Erdős Discrepancy Problem by Lisitsa/Konev
- File containing the proof about 13 GB

- **Lit:** B. Konev and A. Lisitsa. Computer-aided proof of erdos discrepancy properties. Artif. Intell., 224(C):103–118, July 2015.
- **Lit:** `https://rjlipton.wordpress.com/2014/02/28/practically-pnp/`

Definition (Erdős Discrepancy Problem (EDP))

Let (x_n) be a sequence of 1s and 0s and C be a constant. Can one always find positive integers d, k s.t.:

$$|\sum_{i=1}^{k} x_{id}| > C$$

By the way: There are Longer Proofs

Definition (Erdős Discrepancy Problem (EDP))

Let (x_n) be a sequence of 1s and 0s and C be a constant. Can one always find positive integers d, k s.t.: $| \sum_{i=1}^{k} x_{id} | > C$

Illustration:

"A precipice [Steilhang (Germ.)] lies two paces to your left and a pit of vipers two paces to your right. Can you devise a series of steps that will avoid the hazards, even if you are forced to take every second, third or Nth step in your series?"

Lit:

https://www.quantamagazine.org/20151001-tao-erdos-discrepancy-problem/

- Update: There is now an elegant short proof for the full case by mathematician Terence Tao
- **Lit:** The Erdős Discrepancy Problem. arXiv:1509.05363, https://arxiv.org/abs/1509.05363

Desiderata

- Due to the undecidabiltiy result one has to constrain dependencies
- Constraints such that the following are fulfilled:

(C1) Existence of solutions entails existence of universal solutions

(C2) UNIVSOLEXISTENCE decidable and even tractable

(C3) If solutions exists, then universal solutions should be constructible in polynomial time

Chase Helps Again

Theorem

Results of successful chase sequences are universal solutions (and these are sometimes called canonical universal solutions).

Proof Sketch

- Have to show only universality of chase $\mathfrak{T}$
- Use the third definition of universality
- Let $\mathfrak{T}'$ be any solution
- Lemma: Adding facts in chase step preserves homomorphism
  (If $\mathfrak{T}1 \overset{\chi}{\rightsquigarrow} \mathfrak{T}2$ by dependency χ, $\mathfrak{T}3$ fulfills χ and there is $h : \mathfrak{T}1 \overset{hom}{\longrightarrow} \mathfrak{T}3$, then there is $h' : \mathfrak{T}2 \overset{hom}{\longrightarrow} \mathfrak{T}3$)
- Argue inductively starting from empty homomorphism

Nice Properties of Universal Solutions

Theorem

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping where M_τ is the union of egds and weakly acyclic tgds. Then:

- *UNISOLEXISTENCE$_{\mathcal{M}}$ can be solved in PTIME (C2).*
- *And if solutions exist, then a universal solution exists (C1),*
- *and a canonical universal solution can be computed in polynomial time (C3).*

Example (Non-uniqueness of Canonical Universal Solutions)

- $M_{\sigma\tau} = \{\ P(x) \rightarrow \exists y \exists w (E(x,y) \wedge E(x,w))\ \}$
- $M_\tau = \{\ \underbrace{E(x,y) \rightarrow \exists z\ F(y,z)}_{\chi_1}, \underbrace{E(x,y) \wedge E(x,y') \rightarrow y = y'}_{\chi_2}\ \}$
- Source instance $\mathfrak{S} = \{P(a)\}$

- First step: $\mathfrak{T} = \{E(a, \perp_1), E(a, \perp_2)\}$
- Two different solutions
  - Apply χ_1, then χ_2:

$$\mathfrak{T}_1 = \{E(a, \perp_1), F(\perp_1, \perp_3), F(\perp_1, \perp_4)\}$$

  - Apply χ_2, then χ_1:

$$\mathfrak{T}_2 = \{E(a, \perp_1), F(\perp_1, \perp_2)\}$$

Example (Non-uniqueness of Canonical Universal Solutions)

- $M_{\sigma\tau} = \{\ P(x) \to \exists y \exists w (E(x,y) \wedge E(x,w))\ \}$
- $M_\tau = \{\ \underbrace{E(x,y) \to \exists z\ F(y,z)}_{\chi_1}, \underbrace{E(x,y) \wedge E(x,y') \to y = y'}_{\chi_2}\ \}$
- Source instance $\mathfrak{S} = \{P(a)\}$

- First step: $\mathfrak{T} = \{E(a, \perp_1), E(a, \perp_2)\}$
- Two different solutions
  - Apply χ_1, then χ_2:

$$\mathfrak{T}_1 = \{E(a, \perp_1), F(\perp_1, \perp_3)), F(\perp_1, \perp_4)\}$$

  - Apply χ_2, then χ_1:

$$\mathfrak{T}_2 = \{E(a, \perp_1), F(\perp_1, \perp_2)\}$$

Example (Non-uniqueness of Canonical Universal Solutions)

- $M_{\sigma\tau} = \{\ P(x) \rightarrow \exists y \exists w (E(x,y) \wedge E(x,w))\ \}$
- $M_\tau = \{\ \underbrace{E(x,y) \rightarrow \exists z\ F(y,z)}_{\chi_1}, \underbrace{E(x,y) \wedge E(x,y') \rightarrow y = y'}_{\chi_2}\ \}$
- Source instance $\mathfrak{S} = \{P(a)\}$

- First step: $\mathfrak{T} = \{E(a, \perp_1), E(a, \perp_2)\}$
- Two different solutions
  - Apply χ_1, then χ_2:

$$\mathfrak{T}_1 = \{E(a, \perp_1), F(\perp_1, \perp_3)), F(\perp_1, \perp_4)\}$$

  - Apply χ_2, then χ_1:

$$\mathfrak{T}_2 = \{E(a, \perp_1), F(\perp_1, \perp_2)\}$$

Example (Non-uniqueness of Canonical Universal Solutions)

- $M_{\sigma\tau} = \{\ P(x) \rightarrow \exists y \exists w (E(x,y) \wedge E(x,w))\ \}$
- $M_\tau = \{\ \underbrace{E(x,y) \rightarrow \exists z\ F(y,z)}_{\chi_1}, \underbrace{E(x,y) \wedge E(x,y') \rightarrow y = y'}_{\chi_2}\ \}$
- Source instance $\mathfrak{S} = \{P(a)\}$

- First step: $\mathfrak{T} = \{E(a, \perp_1), E(a, \perp_2)\}$
- Two different solutions
  - Apply χ_1, then χ_2:

$$\mathfrak{T}_1 = \{E(a, \perp_1), F(\perp_1, \perp_3)), F(\perp_1, \perp_4)\}$$

  - Apply χ_2, then χ_1:

$$\mathfrak{T}_2 = \{E(a, \perp_1), F(\perp_1, \perp_2)\}$$

Non-uniqueness

- Non-uniqueness no serious problem as all universal solutions are good (for answering CQs)
- Nonetheless one can show

Proposition

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping s.t. M_τ consists of egds only. Then every source instance $\mathfrak{S}$ has a unique canonical solution $\mathfrak{T}$ (up to a renaming of NULLS) under $\mathcal{M}$.

The Core

Example (Core in Flight Domain)

Source schema σ and instance

Geo(city, coun, pop)
paris, france, 2M

Flight (src, dest, airl, dep)
paris amst. klm 1410
paris amst. klm 2230

Target schema τ

Route(fno, src, dest)

Info(fno, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists arr\ (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$

Example (Core in Flight Domain)

Source schema σ and instance

Geo(city,	coun,	pop)
paris,	france,	2M

Flight (src,	dest,	airl,	dep)
paris	amst.	klm	1410
paris	amst.	klm	2230

Target schema τ

Route(fno, src, dest)

Info(fno, dep, arr, airl)

Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

Example (Core in Flight Domain)

Source schema σ and instance

Geo(city,	coun,	pop)
paris,	france,	2M

Flight (src,	dest,	airl,	dep)
paris	amst.	klm	1410
paris	amst.	klm	2230

Target schema τ and universal solution $\mathfrak{T}$

Route(fno,	src,	dest)
$\perp_1$,	paris,	amst.
$\perp_3$,	paris,	amst.

Info(fno,	dep,	arr,	airl)
$\perp_1$,	1410,	$\perp_2$,	klm
$\perp_3$,	2320,	$\perp_4$,	klm

Serves(airl,	city,	coun,	phone)
klm,	paris,	france,	$\perp_5$
klm,	paris,	france,	$\perp_6$

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

Example (Core in Flight Domain)

Source schema σ and instance

Geo(city,	coun,	pop)
paris,	france,	2M

Flight (src,	dest,	airl,	dep)
paris	amst.	klm	1410
paris	amst.	klm	2230

Target schema τ and core solution

Route(fno,	src,	dest)
$\perp_1$,	paris,	amst.
$\perp_3$,	paris,	amst.

Info(fno,	dep,	arr,	airl)
$\perp_1$,	1410,	$\perp_2$,	klm
$\perp_3$,	2320,	$\perp_4$,	klm

Serves(airl,	city,	coun,	phone)
klm,	paris,	france,	$\perp_5$
~~klm,~~	~~paris,~~	~~france,~~	~~$\perp_6$~~

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

Example (Core in Flight Domain)

Source schema σ and instance

Geo(city,	coun,	pop)
paris,	france,	2M

Flight (src,	dest,	airl,	dep)
paris	amst.	klm	1410
paris	amst.	klm	2230

Target schema τ and core solution

Route(fno,	src,	dest)
$\perp_1$,	paris,	amst.
$\perp_3$,	paris,	amst.

Info(fno,	dep,	arr,	airl)
$\perp_1$,	1410,	$\perp_2$,	klm
$\perp_3$,	2320,	$\perp_4$,	klm

Serves(airl,	city,	coun,	phone)
klm,	paris,	france,	$\perp_5$
~~klm,~~	~~paris,~~	~~france,~~	~~$\perp_6$~~

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep)$)

Why not delete similarly $Route(\perp_3, paris, amst)$?

Example (Core in Flight Domain)

Source schema σ and instance

Geo(city,	coun,	pop)
paris,	france,	2M

Flight (src,	dest,	airl,	dep)
paris	amst.	klm	1410
paris	amst.	klm	2230

Target schema τ and core solution

Route(fno,	src,	dest)
$\perp_1$,	paris,	amst.
$\perp_3$,	paris,	amst.

Info(fno,	dep,	arr,	airl)
$\perp_1$,	1410,	$\perp_2$,	klm
$\perp_3$,	2320,	$\perp_4$,	klm

Serves(airl,	city,	coun,	phone)
klm,	paris,	france,	$\perp_5$
~~klm,~~	~~paris,~~	~~france,~~	~~$\perp_6$~~

Mapping rules $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep)$

Why not delete similarly $Route(\perp_3, paris, amst)$?

There are additional facts distinguishing $\perp_1$ and $\perp_3$

Identification $\perp_1 = \perp_3$ would violate primary key constraint

Better than Universal? The Core! (in some sense)

- Universal solutions may still contain redundant information
- Seeking for smallest universal solutions: cores

- $\mathfrak{T}'$ is subinstance of $\mathfrak{T}$, for short $\mathfrak{T}' \subseteq \mathfrak{T}$, iff $R^{\mathfrak{T}'} \subseteq R^{\mathfrak{T}}$ for all relation symbols R

Definition

A subinstance $\mathfrak{T}' \subseteq \mathfrak{T}$ is a core of $\mathfrak{T}$ iff there is $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$ but there is not a homomorphism from $\mathfrak{T}$ to a proper subinstance of $\mathfrak{T}'$.

- Intuitively: An instance can be retracted (structure preservingly) to its core but not further

Properties of Cores

Definition

A subinstance $\mathfrak{T}' \subseteq \mathfrak{T}$ is a core of $\mathfrak{T}$ iff there is $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$ but there is not a homomorphism from $\mathfrak{T}$ to a proper subinstance of $\mathfrak{T}'$.

Proposition

1. *Every instance has a core.*
2. *All cores of the same instance are isomorphic (same up to renaming of NULLs)* *($\Longrightarrow$ Talk of the core justified)*
3. *Two instances are homomorphically equivalent iff their cores are isomorphic*
4. *If $\mathfrak{T}'$ is core of $\mathfrak{T}$, then there is $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$ s.t. $h(\nu) = \nu$ for all $\nu \in DOM(\mathfrak{T}')$*

Main Theorem for Cores

Theorem

1. *If* $\mathfrak{T} \in SOL_{\mathcal{M}}(\mathfrak{S})$, *then also* $core(\mathfrak{T}) \in SOL_{\mathcal{M}}(\mathfrak{S})$

2. *If* $\mathfrak{T} \in UNIVSOL_{\mathcal{M}}(\mathfrak{S})$ *then also* $core(\mathfrak{T}) \in UNIVSOL_{\mathcal{M}}(\mathfrak{S})$

3. *If* $UNIVSOL_{\mathcal{M}}(\mathfrak{S}) \neq \emptyset$, *then all* $\mathfrak{T} \in UNIVSOL_{\mathcal{M}}(\mathfrak{S})$ *have same core (up to renaming of NULLs), and the core of any universal solution is the smallest universal solution*

Computing the Core

- Easy case: No tgds in M_τ
- Simple algorithm $\mathit{COMPUTECORE}(\mathcal{M})$
  - Assume $\mathfrak{S}$ has successful sequence with result $\mathfrak{T}$.
  - If $\mathfrak{T} = \mathit{fail}$, then also the output fail
  - Otherwise: remove facts as long as $M_{\sigma\tau}$ fulfilled.

Theorem

If chase not fails, then $\mathit{COMPUTECORE}(\mathcal{M})$ outputs core of universal solutions in polynomial time.

- Algorithm works as egds satisfactions preserved for subinstances
- More sophisticated methods needed in presence of tgds in M_τ

Core Solution vs. Universal Solution

- Core solutions contain less redundant information and are unique
- but are harder to construct (next lecture)

Core Solution vs. Universal Solution

- Core solutions contain less redundant information and are unique
- but are harder to construct (next lecture)

- Which one to use?
  - Aim "only" answering CQs $\Longrightarrow$ universal solution
  - Aim goes further $\Longrightarrow$ core solution
    - Need to query with more expressive language (negation, counting)
    - Need to calculate sufficient statistics in an ML algorithm

Core Solution vs. Universal Solution

- Core solutions contain less redundant information and are unique
- but are harder to construct (next lecture)

- Which one to use?
  - Aim "only" answering CQs $\Longrightarrow$ universal solution
  - Aim goes further $\Longrightarrow$ core solution
    - Need to query with more expressive language (negation, counting)
    - Need to calculate sufficient statistics in an ML algorithm

- Compare discriminative vs. generative models in ML
  - In discriminative model: Can answer (only) classification query, but easier to construct
  - In generative model: Can answer any (probabilistic) query, but hard to construct

Core Solution vs. Universal Solution

- Core solutions contain less redundant information and are unique
- but are harder to construct (next lecture)

- Which one to use?
  - Aim "only" answering CQs $\Longrightarrow$ universal solution
  - Aim goes further $\Longrightarrow$ core solution
    - Need to query with more expressive language (negation, counting)
    - Need to calculate sufficient statistics in an ML algorithm

- Compare discriminative vs. generative models in ML
  - In discriminative model: Can answer (only) classification query, but easier to construct
  - In generative model: Can answer any (probabilistic) query, but hard to construct

$\Longrightarrow$ Expressivity vs. Feasibility Dilemma

Query Answering

Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$
- Semantics of query answering specified as certain answer semantics

Definition

The certain answers of query Q over τ for given instance $\mathfrak{S}$ is defined as

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) = \bigcap \{\ cert(Q, \mathfrak{T}) \mid \mathfrak{T} \in SOL_{\mathcal{M}}(\mathfrak{S})\ \}$$

Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$
- Semantics of query answering specified as certain answer semantics

Definition

The certain answers of query Q over τ for given instance $\mathfrak{S}$ is defined as

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) = \bigcap \{\ cert(Q, \mathfrak{T}) \mid \mathfrak{T} \in SOL_{\mathcal{M}}(\mathfrak{S})\ \}$$

- Definition does not tell how to actually compute the certain answers
- In many cases it is not necessary to compute all solutions to get certain answers

Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$
- Semantics of query answering specified as certain answer semantics

Definition

The certain answers of query Q over τ for given instance $\mathfrak{S}$ is defined as

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) = \bigcap \{\ cert(Q, \mathfrak{T}) \mid \mathfrak{T} \in SOL_{\mathcal{M}}(\mathfrak{S})\ \}$$

Wake-up Question

Could it be the case that the certain answer set contains NULLS?

Certain Answers

- Given mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$
- Semantics of query answering specified as certain answer semantics

Definition

The certain answers of query Q over τ for given instance $\mathfrak{S}$ is defined as

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) = \bigcap \{ cert(Q, \mathfrak{T}) \mid \mathfrak{T} \in SOL_{\mathcal{M}}(\mathfrak{S}) \}$$

Wake-up Question

Could it be the case that the certain answer set contains NULLS?

Answer: No, because one can construct for any solution another with different NULLs

Algorithmic Problems for Certain Answers

Problem: $CERTAIN_{\mathcal{M}}(Q, \mathfrak{S})$

Input: Source instance $\mathfrak{S}$ and tuple of elements $\vec{t} \in DOM(\mathfrak{S})$
Output: Answer whether $\vec{t} \in certain_{\mathcal{M}}(Q, \mathfrak{S})$

- Again, to guarantee tractability or even decidability one has to restrict the involved components
  - Constrain query language (e.g., from FOL to CQs)
  - Constrain dependencies (e.g., to weakly acyclic TGDs)

Proposition

There is an FOL query Q and a $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ s.t. $CERTAIN_{\mathcal{M}}(Q)$ is undecidable.

Answering Conjunctive Queries (CQs)

- Conjunctive queries (CQs)

$$Q(\vec{x}) = \exists \vec{y} \left(\alpha_1(\vec{x_1}, \vec{y_1}) \wedge \cdots \wedge \alpha_n(\vec{x_n}, \vec{y_n}) \right)$$

- Unions of conjunctive queries (UCQs)

$$Q(\vec{x}) = CQ_1(\vec{x}) \vee \cdots \vee CQ_n(\vec{x})$$

- Crucial Property: (U)CQs are preserved under homomorphisms

Proposition

Let $h : \mathfrak{S} \xrightarrow{hom} \mathfrak{S}'$ and Q be a UCQ. Then: For all tuples $\vec{a}$ from the domain of $\mathfrak{S}$: If $\vec{a} \in Q(\mathfrak{S})$, then $h(\vec{a}) \in Q(\mathfrak{S}')$

If $\mathfrak{S}$ is complete, then the condition boils down to $Q(\mathfrak{S}) \subseteq Q(\mathfrak{S}')$

Follows easily from homomorphism definition (see Exercise)

As a corollary one immediately gets also preservation for certain query answering.

Proposition

Let $h : \mathfrak{S} \xrightarrow{hom} \mathfrak{S}'$ *and* Q *be a UCQ. Then:*

$$cert(Q, \mathfrak{S}) \subseteq cert(Q, \mathfrak{S}')$$

- Here we used the following notion of certain answering for general DBs (independently from a DE scenario)

Definition

$$cert(Q, \mathfrak{S}) = \bigcap \{ Q(\mathfrak{S}') \in Rep(\mathfrak{S}) \}$$

Certain Answers Naively

Definition (Naive evaluation strategy for general DBs)

For an arbitrary general *DB* $\mathfrak{S}$ the set of answers following a naive evaluation strategy, for short $Q_{naive}(\mathfrak{S})$, is calculated as follows

- Treat marked NULLS in $\mathfrak{S}$ as constants (i.e. $\perp = \perp$ is true but not $\perp = c$ or $\perp = \perp'$)
- Calculate $Q(\mathfrak{S})$ under this perspective (treating $\mathfrak{S}$ as ordinary complete DB)
- and then eliminate all tuples from $Q(\mathfrak{S})$ containing a NULL

Theorem

$$cert(\mathfrak{S}, Q) = Q_{naive}(\mathfrak{S})$$

Certain Answers Naively

Theorem

$$cert(\mathfrak{S}, Q) = Q_{naive}(\mathfrak{S})$$

Proof sketch:

- For every $\mathfrak{S}' \in Rep(\mathfrak{S})$ there is $\mathfrak{S} \xrightarrow{hom} \mathfrak{S}'$
- As homomorphism preserves answers of CQs:
  $Q_{naive}(\mathfrak{S}) =$ NULL-free tuples in $Q(\mathfrak{S}) \subseteq \bigcap_{\mathfrak{S}' \in Rep(\mathfrak{S})} Q(\mathfrak{S}')$
- $Q_{naive}(\mathfrak{S}) \supseteq \bigcap_{\mathfrak{S}' \in Rep(\mathfrak{S})} Q(\mathfrak{S}')$ follows from the fact that $\mathfrak{S}$ can be considered as its own completion (when treating nulls consistently as constants).

  Lit: T. Imielinski and W. Lipski, Jr. Incomplete information in relational databases. J. ACM, 31(4):761–791, Sept. 1984.

Certain Answering UCQs

Theorem

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping where M_τ is a union of egds and weakly acyclic tgds and let Q be a UCQ.

Then $CERTAIN_{\mathcal{M}}(Q, \mathfrak{S})$ can be solved in PTIME.

Proof Sketch

- Consider naive evaluation strategy Q_{naive} for source DBs and mappings:
  - Let $\mathfrak{T}$ arbitrarily chosen universal solution
  - Treat marked NULLS in $\mathfrak{T}$ as constants (i.e. $\perp = \perp$ is true but not $\perp = c$ or $\perp = \perp'$)
  - Calculate $Q(\mathfrak{T})$ under this perspective (treating $\mathfrak{T}$ as ordinary complete DB)
  - and then eliminate all tuples from $Q(\mathfrak{T})$ containing a NULL
- Now one can show $certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{naive}(\mathfrak{T})$.

Showing $certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{naive}(\mathfrak{T})$

- We know that a universal solution $\mathfrak{T}$ can be constructed in polynomial time.
- For every $\mathfrak{T}' \in SOL_{\mathcal{M}}$ there is $\mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$
- NULL-free tuples in $Q(\mathfrak{T}) \subseteq$
  $\bigcap_{\mathfrak{T}' \in SOL_{\mathcal{M}}}$ NULL-free tuples in $Q(\mathfrak{T}')$
- Other direction clear, as $\mathfrak{T}$ is a solution by itself

- Answering FOL queries (and so of UCQs) computable in PTIME data complexity

Showing $certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{naive}(\mathfrak{T})$

- We know that a universal solution $\mathfrak{T}$ can be constructed in polynomial time.
- For every $\mathfrak{T}' \in SOL_{\mathcal{M}}$ there is $\mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$
- NULL-free tuples in $Q(\mathfrak{T}) \subseteq$
  $\bigcap_{\mathfrak{T}' \in SOL_{\mathcal{M}}}$ NULL-free tuples in $Q(\mathfrak{T}')$
- Other direction clear, as $\mathfrak{T}$ is a solution by itself

- Answering FOL queries (and so of UCQs) computable in PTIME data complexity

QA for Other Classes of Queries

- Proof above used simple strategy for certain answering by naive evaluation, this time for solutions of a mapping and a source instance $\mathfrak{S}$

Definition (Naive Evaluation Strategy for DEs)

$$cert_{\mathcal{M}}(\mathfrak{S}, Q) = Q_{naive}(\mathfrak{T})$$

where $\mathfrak{T}$ is a universal solutio for $\mathcal{M}$ and $\mathfrak{S}$.

- This strategy works also for Datalog programs as constraints for the target schema τ
  - Reason: Datalog programs are preserved under homomorphisms
  - Even if one adds inequalities, naive evaluation works
  - Hence certain answering is here in PTime

QA for Other Classes of Queries

- Proof above used simple strategy for certain answering by naive evaluation, this time for solutions of a mapping and a source instance $\mathfrak{S}$

Definition (Naive Evaluation Strategy for DEs)

$$cert_{\mathcal{M}}(\mathfrak{S}, Q) = Q_{naive}(\mathfrak{T})$$

where $\mathfrak{T}$ is a universal solutio for $\mathcal{M}$ and $\mathfrak{S}$.

- This strategy works also for Datalog programs as constraints for the target schema τ
  - Reason: Datalog programs are preserved under homomorphisms
  - Even if one adds inequalities, naive evaluation works
  - Hence certain answering is here in PTime

Rewritability

- Naive evaluation is a form of rewriting
- Fundamental method that re-appears in different areas of CS
- Rewrite a query w.r.t. a given KB into a new query that "contains" the knowledge of KB

- Challenges
  - Preserve the semantics in the rewriting process:
    ensure correctness (easy) and completeness (difficult)
  - The language of the output query is constraint to a "simple language" (so rewritability not always guaranteed)

Rewritability for DE

Definition (FOL Rewritability)

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping and Q be a query over τ.

Then Q is said to be FOL-rewritable over the canonical universal solution under $\mathcal{M}$ if there is a FOL query Q_{rew} over τ^C such that

$$certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{rew}(\mathfrak{T})$$

- Here $\tau^C = \tau \cup \{C\}$ where
  unary predicate C depicts all constants (not NULLs) in targets
- Works like a type predicate

Rewritability for DE

Definition (FOL Rewritability)

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping and Q be a query over τ.

Then Q is said to be FOL-rewritable over the canonical universal solution under $\mathcal{M}$ if there is a FOL query Q_{rew} over τ^C such that

$$certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{rew}(\mathfrak{T})$$

One must find one rewriting for any given pair of source $\mathfrak{S}$ and universal solution $\mathfrak{T}$

- The known component is the mapping $\mathcal{M}$
- The unknown components are all pairs $(\mathfrak{S}, \mathfrak{T})$

Rewritability for DE

Definition (FOL Rewritability)

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping and Q be a query over τ.

Then Q is said to be FOL-rewritable over the canonical universal solution under $\mathcal{M}$ if there is a FOL query Q_{rew} over τ^C such that

$$certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{rew}(\mathfrak{T})$$

If, in the definition, one talks about cores $\mathfrak{T}$ instead of universal solutions then Q is said to be FOL rewritable over cores

Theorem

FOL rewrit. over core $\models$ *FOL rewrit. over universal solution, but not vice versa.*

Rewritability for DE

Definition (FOL Rewritability)

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping and Q be a query over τ.

Then Q is said to be FOL-rewritable over the canonical universal solution under $\mathcal{M}$ if there is a FOL query Q_{rew} over τ^C such that

$$certain_{\mathcal{M}}(Q, \mathfrak{S}) = Q_{rew}(\mathfrak{T})$$

Example

- $Q(\vec{x})$: a conjunctive query
- Q_{rew}: $Q(\vec{x}) \wedge C(x_1) \wedge \cdots \wedge C(x_n)$
  This is actually the syntactic form of Q_{naive}
- The rewriting is even independent of $\mathcal{M}$
- So: (U)CQs are rewritable for any mapping

Adding Negations to Query Language

- Negations in query languages lead to lose of naive rewriting technique
- Even if one allows only negation in inequalities

Definition (Conjunctive Queries with inequalities $CQ^{\neq}$)

A conjunctive query with inequalities is a query of the form

$$Q(\vec{x}) = \exists \vec{y} \left(\alpha_1(\vec{x_1}, \vec{y_1}) \wedge \cdots \wedge \alpha_n(\vec{x_n}, \vec{y_n}) \right)$$

where α_i is either an atomic relational formula or an inequality $z_i \neq z_j$.

Example (No Naive Evaluation Possible)

Source DB

Flight (	src,	dest,	airl,	dep)
	paris	sant.	airFr	2320
	paris	sant.	lan	2200

Target DB

Routes(fno, src, dest)

Info(fno, dep, arr, airl)

- **Dependencies** $M_{\sigma\tau}$

$$Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists\ arr(Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

- Any universal solution $\mathfrak{T}'$ contains solution τ **solutions**

$$\mathfrak{T} = \{\ Routes(\perp_1, paris, sant),\ Info(\perp_1, 2320, \perp_2, airFr),\ Routes(\perp_3, paris, sant),\ Info(\perp_3, 2320, \perp_4, lan)\ \}$$

- Query $Q(x, z) = \exists y \exists y'(Routes(y, x, z) \wedge Routes(y', x, z) \wedge y \neq y')$
- $Q_{naive}(\mathfrak{T}') = \{(paris, sant)\}$ (for any universal solution $\mathfrak{T}'$)
- But: $cert(Q(x, z), \mathfrak{S})_{\mathcal{M}} = \emptyset$ because there is a solution

$$\mathfrak{T}'' = \{\ Routes(\perp_1, paris, sant),\ Info(\perp_1, 2320, \perp_2, airFr),$$

Example (No Naive Evaluation Possible)

Source DB

Flight (	src,	dest,	airl,	dep)
	paris	sant.	airFr	2320
	paris	sant.	lan	2200

Target DB

Routes(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

- **Dependencies** $M_{\sigma\tau}$

$$Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists\ arr(Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

- Any universal solution $\mathfrak{T}'$ contains solution τ **solutions**

$$\mathfrak{T} = \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),\ Routes(\bot_3, paris, sant),\ Info(\bot_3, 2320, \bot_4, lan)\ \}$$

- Query $Q(x, z) = \exists y \exists y'(Routes(y, x, z) \wedge Routes(y', x, z) \wedge y \neq y')$
- $Q_{naive}(\mathfrak{T}') = \{(paris, sant)\}$ (for any universal solution $\mathfrak{T}'$)
- But: $cert(Q(x, z), \mathfrak{S})_{\mathcal{M}} = \emptyset$ because there is a solution

$$\mathfrak{T}'' = \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$

Example (No Naive Evaluation Possible)

Source DB

Flight (	src,	dest,	airl,	dep)
	paris	sant.	airFr	2320
	paris	sant.	lan	2200

Target DB

Routes(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

- **Dependencies** $M_{\sigma\tau}$

$$Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists\ arr(Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

- Any universal solution $\mathfrak{T}'$ contains solution τ **solutions**

$$\mathfrak{T} \quad = \quad \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$
$$Routes(\bot_3, paris, sant),\ Info(\bot_3, 2320, \bot_4, lan)\ \}$$

- Query $Q(x, z) = \exists y \exists y'(Routes(y, x, z) \wedge Routes(y', x, z) \wedge y \neq y')$
- $Q_{naive}(\mathfrak{T}') = \{(paris, sant)\}$ (for any universal solution $\mathfrak{T}'$)
- But: $cert(Q(x, z), \mathfrak{S})_{\mathcal{M}} = \emptyset$ because there is a solution

$$\mathfrak{T}'' \quad = \quad \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$

Example (No Naive Evaluation Possible)

Source DB

Flight	src,	dest,	airl,	dep
	paris	sant.	airFr	2320
	paris	sant.	lan	2200

Target DB

Routes(fno, src, dest)

Info(fno, dep, arr, airl)

- **Dependencies** $M_{\sigma\tau}$

$$Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists\ arr(Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

- Any universal solution $\mathfrak{T}'$ contains solution τ **solutions**

$$\mathfrak{T} \quad = \quad \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$
$$Routes(\bot_3, paris, sant),\ Info(\bot_3, 2320, \bot_4, lan)\ \}$$

- Query $Q(x, z) = \exists y \exists y'(Routes(y, x, z) \wedge Routes(y', x, z) \wedge y \neq y')$
- $Q_{naive}(\mathfrak{T}') = \{(paris, sant)\}$ (for any universal solution $\mathfrak{T}'$)
- But: $cert(Q(x, z), \mathfrak{S})_{\mathcal{M}} = \emptyset$ because there is a solution

$$\mathfrak{T}'' \quad = \quad \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$

Example (No Naive Evaluation Possible)

Source DB

Flight (	src,	dest,	airl,	dep)
	paris	sant.	airFr	2320
	paris	sant.	lan	2200

Target DB

Routes(<u>fno</u>, src, dest)

Info(<u>fno</u>, dep, arr, airl)

- **Dependencies** $M_{\sigma\tau}$

$$Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists\ arr(Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

- Any universal solution $\mathfrak{T}'$ contains solution τ **solutions**

$$\mathfrak{T} \quad = \quad \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$
$$Routes(\bot_3, paris, sant),\ Info(\bot_3, 2320, \bot_4, lan)\ \}$$

- Query $Q(x, z) = \exists y \exists y'(Routes(y, x, z) \wedge Routes(y', x, z) \wedge y \neq y')$
- $Q_{naive}(\mathfrak{T}') = \{(paris, sant)\}$ (for any universal solution $\mathfrak{T}'$)
- But: $cert(Q(x, z), \mathfrak{S})_{\mathcal{M}} = \emptyset$ because there is a solution

$$\mathfrak{T}'' \quad = \quad \{\ Routes(\bot_1, paris, sant),\ Info(\bot_1, 2320, \bot_2, airFr),$$

$CQ^{\neq}$ is in coNP

- In case of $CQ^{\neq}$ one cannot even find a tractable means to answer them w.r.t. certain answer semantics

Theorem

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping where M_τ is the union of egds and weakly acyclic tgds, and let Q be a $UCQ^{\neq}$ query. Then:

$CERTAIN_{\mathcal{M}}(Q)$ is in coNP

Non-rewritability

- Generally it is not possible to decide whether rewritability holds

Theorem

For mappings without target constraints one can not decide whether a given FOL query is rewritable over the canonical solutions (over the core).

- Showing Non-FOL-rewritability can be done with locality tools
- Actually: One uses Hanf-locality of FOL
- Adaptation to DE setting

Not Covered

- Different semantics for query answering
  - Combinations of open-world (certain answers) and closed-word semantics
- Whole sub-field of mapping management
  - How to compose mappings
  - How to maintain mappings (e.g., w.r.t. consistency)
  - How to invert mappings: Get back source DB from target DB
- DE for non-relational DBs
  - e.g., DE for semi-structured data (XML)
  - techniques other than that for relation DE needed

Solutions to Exercise 5 (16 points)

Exercise 5.1 (6 Points)

A tuple-generating dependency (tgd) is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, \psi(\vec{x},\vec{z}))$$

where ϕ, ψ are conjunctions of atoms over τ.

1. Foxy asserts that one can also use existentials in the body (that is in the part before the $\longrightarrow$ without expressivity. Is he right? Argue for your answer!
2. Smarty asserts that one can safely assume that ψ is just an atom rather than a conjunction of atoms. Is he right? Argue for your answer!
3. Dumby asserts similarly that one can safely assume that ϕ is just an atom rather than a conjunction of atoms. Is he right? Argue for your answer!

Solution to Exercise 5.1.1

- Foxy is right.
- One uses the equivalence

$$\forall x(F(x) \to B) \equiv (\exists x F(x)) \to B$$

where x does not appear in B

Solution to Exercise 5.1.2

- Smarty is right
- Let the tgd be of the form

$$\forall \vec{x}\vec{y}(\phi(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, R_1(\vec{x},\vec{z}) \wedge \cdots \wedge R_n(\vec{x},\vec{z})$$

- Introduce new relation $S(\vec{x},\vec{z})$.
- Then the tgd can be replaced by the set of tgds

$$\begin{aligned} \forall \vec{x}\vec{y}(\phi(\vec{x},\vec{y}) &\longrightarrow \exists \vec{z}\, S(\vec{x},\vec{z}) \\ S(\vec{x},\vec{z}) &\longrightarrow R_1(\vec{x},\vec{z}) \\ S(\vec{x},\vec{z}) &\longrightarrow R_2(\vec{x},\vec{z}) \\ &\cdots \\ S(\vec{x},\vec{z}) &\longrightarrow R_n(\vec{x},\vec{z}) \end{aligned}$$

- This set is at least satisfiably equivalent to the original tgd.

Solution to Exercise 5.1.3

- Dumby is wrong
- Consider just the propositional variant $a \wedge b \rightarrow c$.
- One cannot find a (finite) set of rules of the form $p \rightarrow q$ that, e.g., leads to chases which are the same w.r.t. a, b
- Consider input DB $\mathfrak{S} = \{a, b\}$. Then $a \wedge b \rightarrow c$ leads to the chase $\{a, b, c\}$.
- If there were an equivalent set E of rules of the form $p \rightarrow q$, then one would have to have a (smallest) path of rule applications were the first rule is of the form $a \rightarrow q$ (case 1) or $b \rightarrow q$ (case 2) and the last rule is of the form $p \rightarrow c$. But then E would lead to a chase containing $\{a, b, c\}$ on $\{a\}$ (for case 1) or $\{b\}$ (case 2) whereas this is not the case for $a \wedge b \rightarrow c$ on $\{a\}$ or $\{b\}$.

Exercise 5.2 (4 Points)

Prove the folklore proposition that conjunctive queries are preserved under homomorphisms, i.e., show that if there is a homomorphism h from a DB instance $\mathfrak{T}$ to a DB instance $\mathfrak{T}'$, then for any CQ $\phi(\vec{x})$:

$$\{h(\vec{d}) \mid \vec{d} \in ans(\phi(\vec{x}), \mathfrak{T})\} \subseteq ans(\phi(\vec{x}), \mathfrak{T}')$$

Solution

- Let $\phi(\vec{x}) = \exists \vec{y} \bigwedge R_i(\vec{x}, \vec{y})$
- Let $\vec{d} \in \vec{d} \in ans(\phi(\vec{x}), \mathfrak{T})$.
- Hence there are $\vec{e}$ s.t. all $R_i(\vec{d}, \vec{e})$ are contained in $\mathfrak{T}$. Due to the homomorphism condition we have that all $R_i(h(\vec{d}), (\vec{e}))$ are in $\mathfrak{T}'$. Hence $h(\vec{d}) \in ans(\phi(\vec{x}), \mathfrak{T}')$.

Exercise 5.3 (6 Points)

1. Testing subsumption (= containment) of CQs is in NP. What about the subsumption of arbitrary FOLs?
2. Show that every CQ (under the usual set semantics) is monotonic and satisfiable.
3. The semantics of CQs is given by the usual FOL set semantics. Inform yourself on the so-called multi-set semantics (aka: bag semantics) for queries and answer the following questions:
   - 3.1 Why should one consider multi-set semantics—in particular if one is interested in SQL queries?
   - 3.2 What does one know about the (complexity of the) subsumption test for CQs under multi-set semantics?

Solution to Exercise 5.3

1. Not decidable. Otherwise one could decide validity of FOL formulae ϕ: Testing for $\models \phi$ can be reduced to testing whether $\top$ is subsumed by ϕ .
2. Satisfiability: Construct canonical DB $\mathfrak{S}_\phi$ associated with ϕ by considering the atoms in ϕ as a facts in DB (treating variables as constants)
   Monotonicity: Any answer over a DB $\mathfrak{S}$ to a CQ ϕ is a special homomorphism h from $\mathfrak{S}_\phi$ into $\mathfrak{S}$. If $\mathfrak{S} \subseteq \mathfrak{S}'$, then h is also a homomorphism into $\mathfrak{S}'$ and hence an answer to the CQ over $\mathfrak{S}'$.
3. Multi-sets
   - 3.1 In SQL naturally multi-sets arise (say by doing a projection)
   - 3.2 One does not even no (yet) whether it is decidable. It has been shown to be Π_p^2. (= class in so-called polynomial hierarchy, $NP \leq \Pi_p^2 \leq PSAPCE$.

   For more information on the query containment problem see nice keynote of Kolaitis:
   Lit: P. Kolaitis. The query containment problem: Set semantics vs. bag semantics. Keynote talk at the Alberto Mendelzon International Workshop on Foundations of Data Management 2013, May 2013.
   `http://ceur-ws.org/Vol-1087/keynote2slides.pdf`

Exercise 6 (10 + 6 points)

Exercise 6.1 (6 bonus points)

Show the following technical lemma:

(*) If $\mathfrak{T} \in Sol_{\mathcal{M}}(\mathfrak{S})$, then also $\mathfrak{T}' \in Sol_{\mathcal{M}}(\mathfrak{S})$, where $\mathfrak{T}'$ results from $\mathfrak{T}$ by substituting all marked nulls $\perp_1, \ldots, \perp_n$ occurring in $\mathfrak{T}$ (consistently) with new constants c_i.

Exercise 6.2 (4 points)

Show that the first definition of universal solutions $USol_1(\mathfrak{T})$ entails the second definition of universal solutions $USol_2(\mathfrak{T})$. Lemma (*) from Exercise 6.1 may be helpful.

Exercise 6.3 (6 Points)

1. Prove that every finite graph has a core (2 points)
2. Prove that two cores of the same graph are isomorphic. (4 points)