



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Data Exchange 1

Lecture 6: Incomplete DBs, universal solutions, core, chase
14 May 2020

Informationssysteme CS4130
(Summer 2020)

References



Lit: M. Arenas, P. Barceló, L. Libkin, and F. Murlak: Foundations of Data Exchange. Cambridge University Press, 2014.

Data Exchange: Motivation

Data Exchange History

- ▶ Much research in DB community

- ▶ Formal treatment starts in 2003

Lit: R. Fagin et al. Data exchange: Semantics and query answering. In: Database Theory - ICDT 2003, Proceedings, volume 2572 of LNCS, pages 207–224. Springer, 2003.

Lit: R. Fagin, L. M. Haas, M. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. Conceptual modeling: Foundations and applications. chapter Clio: Schema Mapping Creation and Data Exchange, pages 198–236. Springer-Verlag, Berlin, Heidelberg, 2009.

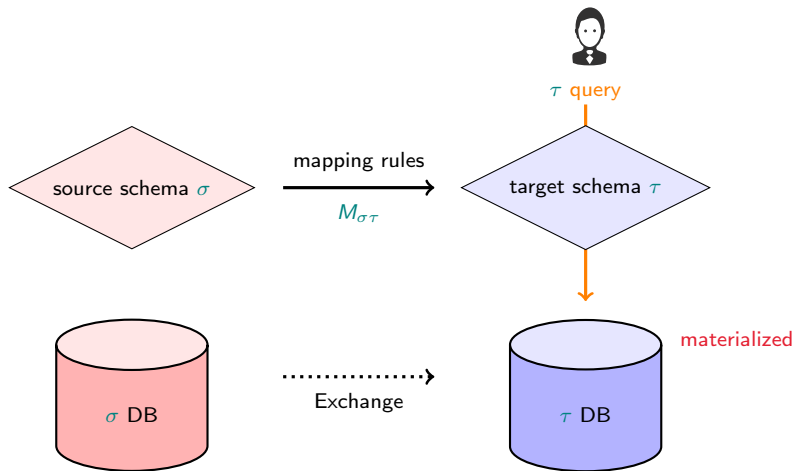
- ▶ Incorporated into IBM Clio

<http://dblab.cs.toronto.edu/project/clio/>

- ▶ A non-commercial DE system:

<http://www.db.unibas.it/projects/llunatic/>

Data Exchange (DE): Main Setting



DE systems

- ▶ DE system $(\sigma, \tau, M_{\sigma\tau}, M_{\tau}) = \mathcal{DI}$ sytem
- ▶ A DE scenario = DE system + source (σ) DB-instance

DE systems

- ▶ DE system $(\sigma, \tau, M_{\sigma\tau}, M_{\tau}) = \mathcal{DI}$ sytem
- ▶ A DE scenario = DE system + source (σ) DB-instance

DE systems

- ▶ DE system $(\sigma, \tau, M_{\sigma\tau}, M_{\tau}) = \mathcal{DI}$ sytem
- ▶ A DE scenario = DE system + source (σ) DB-instance
- ▶ We call here a DE system also a (relational) mapping \mathcal{M} (following Arenas et al. 2014)

DE systems

- ▶ DE system $(\sigma, \tau, M_{\sigma\tau}, M_{\tau}) = \mathcal{DI}$ system
- ▶ A DE scenario = DE system + source (σ) DB-instance
- ▶ We call here a DE system also a (relational) mapping \mathcal{M} (following Arenas et al. 2014)
- ▶ We will deal in detail with target constraints M_{τ} for DE systems (similar treatment for DI scenarios in virtual mode)

Relational Mappings Formally

Definition

A relational mapping \mathcal{M} is a tuple of the form

$$\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$$

where

- ▶ σ is the source schema
- ▶ τ is the target schema with all relation symbols different from those in σ
- ▶ $M_{\sigma\tau}$ is a finite set of FOL formulae over $\sigma \cup \tau$ called **source-to-target dependencies**
 - ▶ As in \mathcal{DI} will consider source-to-target tuple generating dependencies (see lecture on \mathcal{DI})
 - ▶ But: In DE (according to Arenas et al. 2014) exact rules not considered: rules always from sources (body) to target (head)
- ▶ M_{τ} is a set of constraints on the target schema called **target dependencies**

Target Dependencies M_τ

- ▶ These define constraints on target schema known also from classical DB theory
- ▶ Two different types of dependencies are sufficiently general to capture the classical DB constraints

Definition

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

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

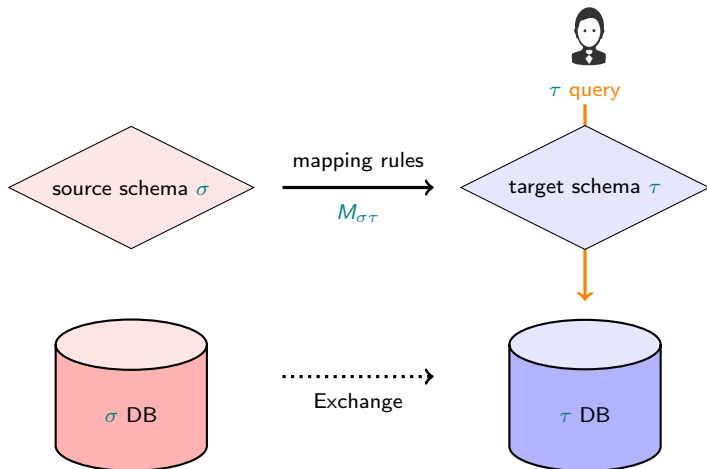
where ϕ, ψ are conjunctions of atoms over τ .

An **equality-generating (egd)** is a FOL formula of the form

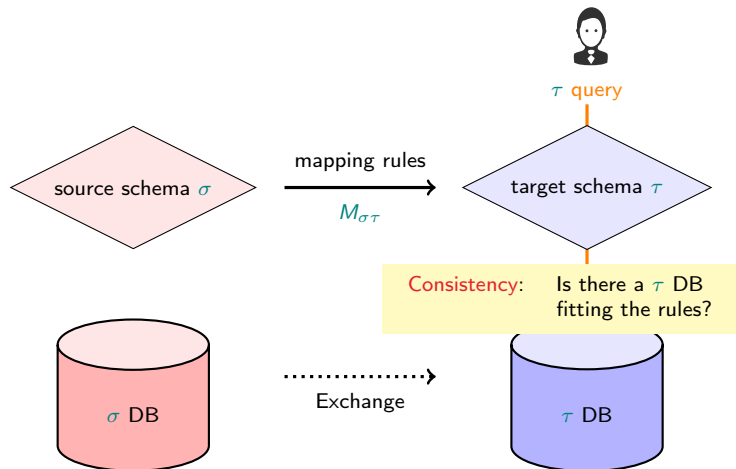
$$\forall \vec{x} (\phi(\vec{x}) \longrightarrow x_i = x_j)$$

where $\phi(\vec{x})$ is a conjunction of atoms over τ and x_i, x_j occur in \vec{x} .

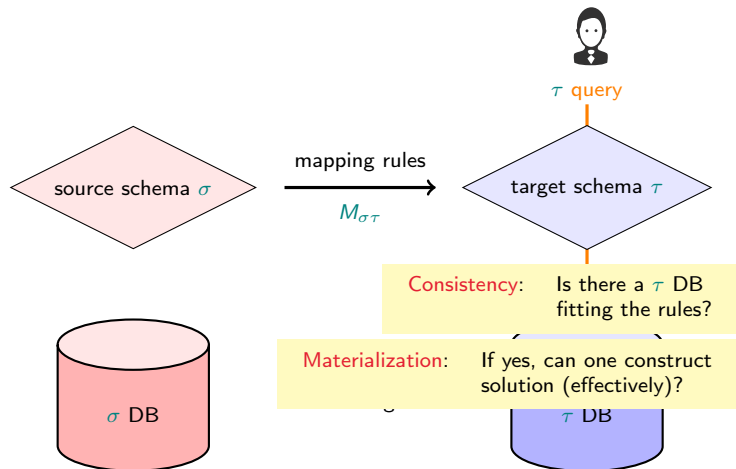
Data Exchange: Challenges



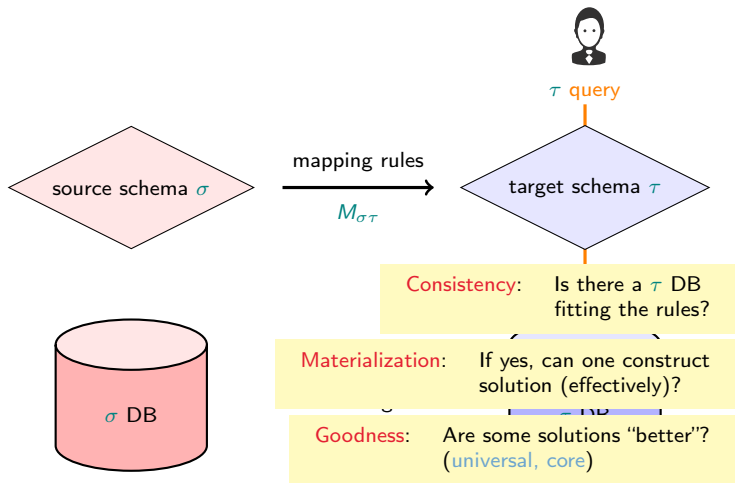
Data Exchange: Challenges



Data Exchange: Challenges

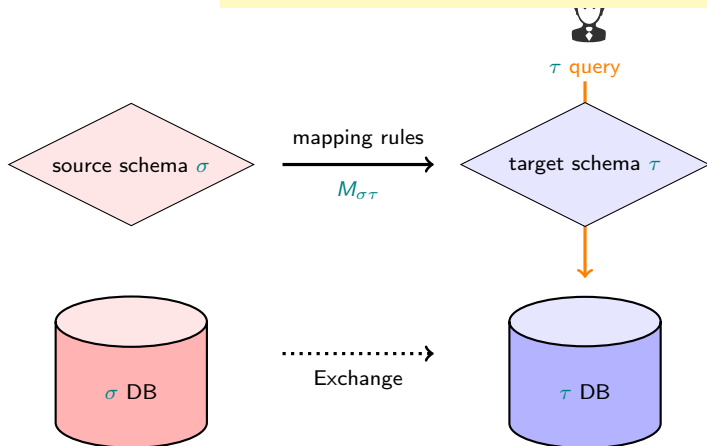


Data Exchange: Challenges



Data Exchange: Challenges

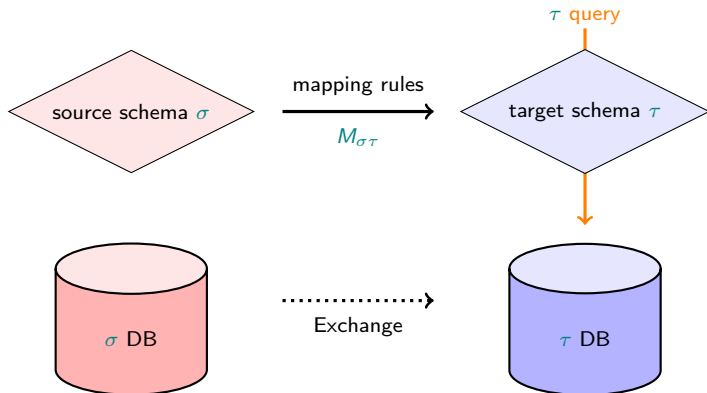
Semantics of QA: What answers are intended?
certain answers



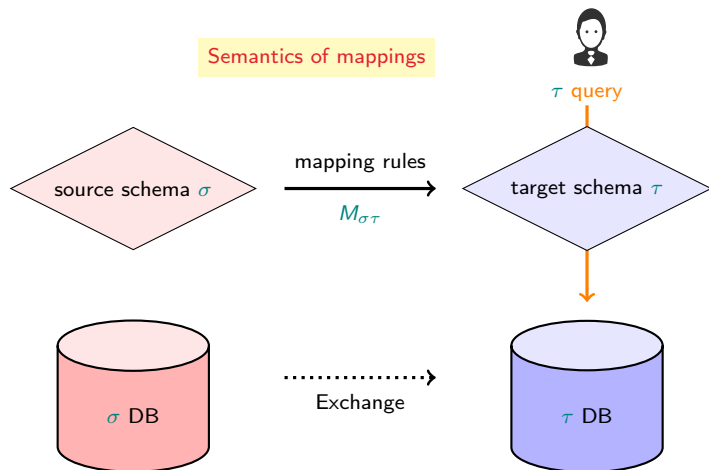
Data Exchange: Challenges

Semantics of QA: What answers are intended?
certain answers

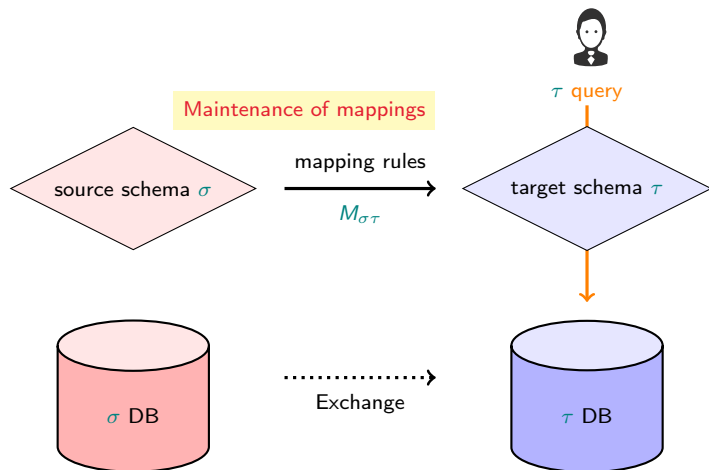
QA algorithm: How to calculate answers?



Data Exchange: Challenges



Data Exchange: Challenges



Running Example

Example (DE in Flight Domain)

Source schema σ

Geo(city, coun, pop)
Flight(src, dest, airl, dep)

Target schema τ

Route(fno, src, dest)
Info(fno, dep, arr, airl)
Serves(airl, city, coun, phone)

Example (DE in Flight Domain)

Source schema σ

Geo(city, coun, pop)
Flight(src, dest, airl, dep)

Target schema τ

Route(fno, src, dest)
Info(fno, dep, arr, airl)
Serves(airl, city, coun, phone)

Target constraints M_τ

primary key: fno

foreign key: $Info[\underline{fno}] \subseteq_{FK} Route[\underline{fno}]$

Example (DE in Flight Domain)

Source schema σ

Geo(city, coun, pop)
Flight(src, dest, airl, dep)

Target schema τ

Route(fno, src, dest)
Info(fno, dep, arr, airl)
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

Example (DE in Flight Domain)

Source schema σ

Geo(city, coun, pop)
Flight(src, dest, airl, dep)

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. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone (Serves(airl, city, coun, phone))$

$M_{\sigma\tau}$ = source-to-target tuple generating dependencies

Sufficiently expressive FOL formula of feasible form
(see lecture on *DI*)

Example (DE in Flight Domain)

Source schema σ and instance \mathcal{G}

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

Source schema σ and instance \mathfrak{G}

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

Source schema σ and instance \mathcal{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_2 , airFr
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. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone (Serves(airl, city, coun, phone))$

Solutions and Certain Answers

Example (DE in Flight Domain)

Source schema σ and instance \mathcal{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Materialization of a τ instance (τ solution)

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

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 and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

Materialization of a τ instance (τ solution)

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

- Non-complete DB: contains marked NULLs \perp_1, \perp_2

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

Materialization of a τ instance (τ solution)

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

- ▶ Non-complete DB: contains marked NULLs \perp_1, \perp_2
- ▶ Can answer τ -queries on \mathfrak{I} using Certain-Answer Semantics

DB Instances of Schemata

- ▶ Schemata are relational signatures
- ▶ Concrete/Complete database instance
 - ▶ For a given schema σ a concrete DB instance is a σ FOL structure with active domain
 - ▶ Active domain: Domain contains all and only individuals (also called constants) occurring in relations
 - ▶ Usually: All source instances are concrete DBs

DB Instances of Schemata

- ▶ Schemata are relational signatures
- ▶ Concrete/Complete database instance
 - ▶ For a given schema σ a concrete DB instance is a σ FOL structure with active domain
 - ▶ Active domain: Domain contains all and only individuals (also called constants) occurring in relations
 - ▶ Usually: All source instances are concrete DBs
- ▶ Generalized/Incomplete DB instances
 - ▶ For some attributes in target schema no corresponding attribute in source may exist (Example: flight number fno)
 - ▶ Next to constants CONST allow disjoint set VAR of marked NULLs
 - ▶ May contain elements from $\text{CONST} \cup \text{VAR}$

Definition (Certain answers over incomplete DB (informally))

$cert(Q, \mathcal{I})$ = intersection of answers over all complete DBs represented by \mathcal{I}

Definition (Certain answers over incomplete DB (informally))

$cert(Q, \mathcal{I})$ = intersection of answers over all complete DBs represented by \mathcal{I}

$Rep(\mathcal{I})$ = all complete DBs resulting from \mathcal{I} by substituting marked NULLs (consistently) with constants

Definition (Certain answers over incomplete DB (formally))

$$\text{cert}(Q, \mathcal{I}) = Q(\mathcal{I}) = \bigcap_{\mathcal{I}' \in \text{Rep}(\mathcal{I})} Q(\mathcal{I}')$$

$\text{Rep}(\mathcal{I})$ = all complete DBs resulting from \mathcal{I} by substituting marked NULLs (consistently) with constants

Definition (Certain answers over incomplete DB (formally))

$$\text{cert}(Q, \mathcal{T}) = Q(\mathcal{T}) = \bigcap_{\mathcal{T}' \in \text{Rep}(\mathcal{T})} Q(\mathcal{T}')$$

$\text{Rep}(\mathcal{T})$ = all complete DBs resulting from \mathcal{T} by substituting marked NULLs (consistently) with constants

Example (Answer for τ solution from flight domain)

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

$$\begin{aligned} \text{Rep}(\mathcal{T}) = & \{ \{ \text{Route}(123, \text{paris}, \text{sant}), \text{Info}(123, 2320, 0815, \text{airFr}) \}, \\ & \{ \{ \text{Route}(124, \text{paris}, \text{sant}), \text{Info}(124, 2320, 0915, \text{airFr}) \}, \\ & \dots, \} \end{aligned}$$

Definition (Certain answers over incomplete DB (formally))

$$\text{cert}(Q, \mathcal{T}) = Q(\mathcal{T}) = \bigcap_{\mathcal{T}' \in \text{Rep}(\mathcal{T})} Q(\mathcal{T}')$$

$\text{Rep}(\mathcal{T})$ = all complete DBs resulting from \mathcal{T} by substituting marked NULLs (consistently) with constants

Example (Answer for τ solution from flight domain)

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

$$\begin{aligned} \text{Rep}(\mathcal{T}) = & \{ \{ \text{Route}(123, \text{paris}, \text{sant}), \text{Info}(123, 2320, 0815, \text{airFr}) \}, \\ & \{ \{ \text{Route}(124, \text{paris}, \text{sant}), \text{Info}(124, 2320, 0915, \text{airFr}) \}, \\ & \dots, \} \end{aligned}$$

► $Q_1 = \exists fno \text{Route}(fno, \text{paris}, \text{sant})$

$$\text{cert}(Q_1, \mathcal{T}) = \{()\} = \text{yes}$$

Definition (Certain answers over incomplete DB (formally))

$$\text{cert}(Q, \mathcal{T}) = Q(\mathcal{T}) = \bigcap_{\mathcal{T}' \in \text{Rep}(\mathcal{T})} Q(\mathcal{T}')$$

$\text{Rep}(\mathcal{T})$ = all complete DBs resulting from \mathcal{T} by substituting marked NULLs (consistently) with constants

Example (Answer for τ solution from flight domain)

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

$$\begin{aligned} \text{Rep}(\mathcal{T}) = & \{ \{ \text{Route}(123, \text{paris}, \text{sant}), \text{Info}(123, 2320, 0815, \text{airFr}) \}, \\ & \{ \{ \text{Route}(124, \text{paris}, \text{sant}), \text{Info}(124, 2320, 0915, \text{airFr}) \}, \\ & \dots, \} \end{aligned}$$

▶ $Q_1 = \exists fno \text{Route}(fno, \text{paris}, \text{sant})$

$$\text{cert}(Q_1, \mathcal{T}) = \{ \{ \} \} = \text{yes}$$

▶ $Q_2 = \text{Route}(123, \text{paris}, \text{sant})$

$$\text{cert}(Q_2, \mathcal{T}) = \emptyset = \text{no}$$

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{G}

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

Target schema τ and instance

Route(fno, src, dest)
Info(fno, dep, arr, airl)
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Example (DE in Flight Domain)

Source schema σ and instance \mathcal{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Many τ solutions $SOL_{\mathcal{M}}(\mathcal{G})$

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

Example (DE in Flight Domain)

Source schema σ and instance \mathcal{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_3 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_3 , 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Many τ solutions $SOL_{\mathcal{M}}(\mathcal{G})$

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

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

Example (DE in Flight Domain)

Source schema σ and instance \mathcal{G}

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

Target schema τ and instance

Route(fno, src, dest)
 \perp_1 , paris, sant.
Info(fno, dep, arr, airl)
 \perp_1 , 2320, \perp_1 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Many τ solutions $SOL_{\mathcal{M}}(\mathcal{G})$

$$\begin{aligned}\mathcal{T} &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\} \\ \mathcal{T}' &= \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\} \\ \mathcal{T}'' &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}\end{aligned}$$

Example (DE in Flight Domain)

Source schema σ and instance \mathcal{G}

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

Target schema τ and instance

Route(fno, src, dest)
 123, paris, sant.
Info(fno, dep, arr, airl)
 123, 2320, \perp_2 , airFr
Serves(airl, city, coun, phone)

Mapping rules $M_{\sigma\tau}$

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

...

Many τ solutions $SOL_{\mathcal{M}}(\mathcal{G})$

$$\begin{aligned}\mathcal{T} &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\} \\ \mathcal{T}' &= \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\} \\ \mathcal{T}'' &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\} \\ \mathcal{T}''' &= \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots\end{aligned}$$

Certain Answers Help Again

- ▶ Same idea: Compromise over all possible solutions
- ▶ Certain answers w.r.t. mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ and source DB \mathcal{G}

Certain Answers Help Again

- ▶ Same idea: Compromise over all possible solutions
- ▶ Certain answers w.r.t. mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ and source DB \mathfrak{G}

Definition (Certain answers (formally))

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) = \bigcap_{\mathfrak{T} \in \text{Sol}_{\mathcal{M}}(\mathfrak{G})} Q(\mathfrak{T})$$

Certain Answers Help Again

- ▶ Same idea: Compromise over all possible solutions
- ▶ Certain answers w.r.t. mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ and source DB \mathfrak{G}

Definition (Certain answers (formally))

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) = \bigcap_{\mathfrak{T} \in \text{Sol}_{\mathcal{M}}(\mathfrak{G})} Q(\mathfrak{T})$$

Example (Certain answers in flight domain)

$$\begin{aligned}\mathfrak{T} &= \{Route(\perp_1, \text{paris}, \text{sant}), \text{Info}(\perp_1, 2320, \perp_2, \text{airFr})\} \\ \mathfrak{T}' &= \{Route(\perp_3, \text{paris}, \text{sant}), \text{Info}(\perp_3, 2320, \perp_2, \text{airFr})\} \\ \mathfrak{T}'' &= \{Route(\perp_1, \text{paris}, \text{sant}), \text{Info}(\perp_1, 2320, \perp_1, \text{airFr})\} \\ \mathfrak{T}''' &= \{Route(123, \text{paris}, \text{sant}), \text{Info}(123, 2320, \perp_2, \text{airFr})\} \dots\end{aligned}$$

- ▶ $Q_1 = \exists fno \text{Route}(fno, \text{paris}, \text{sant})$ $\text{cert}_{\mathcal{M}}(Q_1, \mathfrak{G}) = \{()\} = \text{yes}$
- ▶ $Q_2 = \text{Route}(123, \text{paris}, \text{sant})$ $\text{cert}_{\mathcal{M}}(Q_2, \mathfrak{G}) = \emptyset = \text{no}$

Good Solutions

One solution to rule them all ...

- ▶ In DE one aims at materializing **exactly one** τ solution!
- ▶ Is there a single solution \mathfrak{T}_u capturing the certain answers?

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

One solution to rule them all ...

- ▶ In DE one aims at materializing **exactly one** τ solution!
- ▶ Is there a single solution \mathfrak{T}_u capturing the certain answers?

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- ▶ Yes! **Universal solution**

One solution to rule them all ...

- ▶ In DE one aims at materializing **exactly one** τ solution!
- ▶ Is there a single solution \mathfrak{T}_u capturing the certain answers?

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- ▶ Yes! **Universal solution**
 - ▶ Contains facts which are as specific as necessary, i.e., all other solutions more specific

One solution to rule them all ...

- ▶ In DE one aims at materializing **exactly one** τ solution!
- ▶ Is there a single solution \mathfrak{T}_u capturing the certain answers?

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- ▶ Yes! **Universal solution**
 - ▶ Contains facts which are as specific as necessary, i.e., all other solutions more specific
 - ▶ Works for CQs = conjunctive queries (= SPJ fragment)

One solution to rule them all ...

- ▶ In DE one aims at materializing **exactly one** τ solution!
- ▶ Is there a single solution \mathfrak{T}_u capturing the certain answers?

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- ▶ Yes! **Universal solution**
 - ▶ Contains facts which are as specific as necessary, i.e., all other solutions more specific
 - ▶ Works for CQs = conjunctive queries (= SPJ fragment)
 - ▶ Universality fundamental property ubiquitous in CS
 - ▶ e.g., most general unifier in resolution

One solution to rule them all ...

- ▶ In DE one aims at materializing **exactly one** τ solution!
- ▶ Is there a single solution \mathfrak{T}_u capturing the certain answers?

$$\text{cert}_{\mathcal{M}}(Q, \mathfrak{G}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- ▶ Yes! **Universal solution**
 - ▶ Contains facts which are as specific as necessary, i.e., all other solutions more specific
 - ▶ Works for CQs = conjunctive queries (= SPJ fragment)
 - ▶ Universality fundamental property ubiquitous in CS
 - ▶ e.g., most general unifier in resolution
 - ▶ If existent, can be constructed by **chase** procedure

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{I}

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

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

$$\mathfrak{I}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

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

$$\mathfrak{I}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots$$

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{I}

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

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

$$\mathfrak{I}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

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

$$\mathfrak{I}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots$$

non-necessary
co-reference

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{I}

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

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

$$\mathfrak{I}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

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

$$\mathfrak{I}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\}$$

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(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

$$\begin{aligned}\mathfrak{U} &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\} \\ \mathfrak{U}' &= \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\} \\ \mathfrak{U}'' &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\} \\ \mathfrak{U}''' &= \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots\end{aligned}$$

Why is \mathfrak{U} 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

$$\begin{aligned} \perp_1 \mapsto \perp_3 \left(\begin{array}{l} \mathfrak{S} \\ \mathfrak{S}' \end{array} \right) &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\} \\ &= \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\} \\ \mathfrak{S}'' &= \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\} \\ \mathfrak{S}''' &= \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots \end{aligned}$$

Why is \mathfrak{S} universal?

Example (DE in Flight Domain)

Source schema σ and instance \mathfrak{I}

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

$$\perp_2 \mapsto \perp_1 \left(\begin{array}{l} \mathfrak{I} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\} \\ \mathfrak{I}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\} \\ \mathfrak{I}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\} \\ \mathfrak{I}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots \end{array} \right.$$

Why is \mathfrak{I} 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 $M_{\sigma\tau}$

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

Universal solutions

$$\begin{array}{l} \perp_1 \mapsto 123 \\ \left. \begin{array}{l} \mathfrak{S} \\ \mathfrak{S}' \\ \mathfrak{S}'' \\ \mathfrak{S}''' \end{array} \right\} = \begin{array}{l} \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\} \\ \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\} \\ \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\} \\ \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \dots \end{array} \end{array}$$

Why is \mathfrak{S} 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

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

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

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

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

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

Homomorphism

- ▶ $CONST(\mathcal{T})$ = set of all constants in \mathcal{T}
- ▶ $VAR(\mathcal{T})$ = set of all marked nulls in \mathcal{T}

Homomorphism

- ▶ $CONST(\mathfrak{T})$ = set of all constants in \mathfrak{T}
- ▶ $VAR(\mathfrak{T})$ = set of all marked nulls in \mathfrak{T}

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 $R(t_1, \dots, t_n) \in \mathfrak{T}$, then $R(h(t_1), \dots, h(t_n)) \in \mathfrak{T}'$
(for all relations R)

Definition (Universal Solution)

A solution \mathcal{T} for \mathcal{G} and \mathcal{M} is called a **universal solution** iff it can be mapped homomorphically into all other solutions.

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

Definition (Universal Solution)

A solution \mathcal{T} for \mathcal{G} and \mathcal{M} is called a **universal solution** iff it can be mapped homomorphically into all other solutions.

For all $\mathcal{T}' \in \text{SOL}_{\mathcal{M}}(\mathcal{G})$ there is $h : \mathcal{T} \xrightarrow{\text{hom}} \mathcal{T}'$

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

Proposition

Let $h : \mathcal{G} \xrightarrow{\text{hom}} \mathcal{G}'$ and Q be a (U)CQ. Then: For all tuples \vec{a} from the domain of \mathcal{G} :

- ▶ If $\vec{a} \in Q(\mathcal{G})$, then $h(\vec{a}) \in Q(\mathcal{G}')$
- ▶ If \mathcal{G} is complete, then even $Q(\mathcal{G}) \subseteq Q(\mathcal{G}')$
- ▶ Corollary: $\text{cert}(Q, \mathcal{G}) \subseteq \text{cert}(Q, \mathcal{G}')$

Example (Non-existence of Universal Solutions)

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

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

Proof sketch (by contradiction)

- ▶ A universal solution \mathfrak{T} must have an infinite sequence $(\mathfrak{G}, \{ G(a, b), L(b, \nu_1), G(\nu_1, \nu_2), L(\nu_2, \nu_3), G(\nu_3, \nu_4) \dots \})$
- ▶ 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_j
- ▶ There must be an $h : \mathfrak{T} \xrightarrow{\text{hom}} \mathfrak{T}'$.
- ▶ But then $h(\nu_i) = c_i$ and hence $h(\nu_{2i-1}) = c_{2i-1}$, but also $h(\nu_{2i-1}) = h(a) = a$, so $c_{2i-1} = a$, ⚡

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 $M_{\sigma\tau}$

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

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{I}

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 $M_{\sigma\tau}$

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

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(<u>fno</u> , src, dest)			
\perp_1 , paris, amst.			
\perp_3 , paris, amst.			
Info(<u>fno</u> , 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 $M_{\sigma\tau}$

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

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(<u>fno</u> , src, dest)
\perp_1 , paris, amst.
\perp_3 , paris, amst.
Info(<u>fno</u> , 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
- ▶ \mathcal{T}' is **subinstance** of \mathcal{T} , for short $\mathcal{T}' \subseteq \mathcal{T}$, iff $R^{\mathcal{T}'} \subseteq R^{\mathcal{T}}$ for all relation symbols R

Definition

A subinstance $\mathcal{T}' \subseteq \mathcal{T}$ is a **core** of \mathcal{T} iff there is $h : \mathcal{T} \xrightarrow{\text{hom}} \mathcal{T}'$ but there is no homomorphism from \mathcal{T} to a proper subinstance of \mathcal{T}' .

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

Properties of Cores

Definition

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

Proposition

1. *Every instance has a core.*
2. *All cores of the same instance are isomorphic (same up to renaming of NULLs) $(\implies$ Talk of **the** core justified)*
3. *Two instances are homomorphically equivalent (= there exists a homomorphism from one into the other and vice versa) iff their cores are isomorphic*
4. *If \mathcal{T}' is core of \mathcal{T} , then there is $h : \mathcal{T} \xrightarrow{\text{hom}} \mathcal{T}'$ s.t. $h(\nu) = \nu$ for all $\nu \in \text{DOM}(\mathcal{T}')$*

Core Solution vs. Universal Solution

- ▶ Core solutions contain less redundant information and are unique
- ▶ but are harder to construct

Core Solution vs. Universal Solution

- ▶ Core solutions contain less redundant information and are unique
- ▶ but are harder to construct

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

Testing for and Constructing Solutions

Reminder: Solutions

Definition

Given: a mapping \mathcal{M} and a σ instance \mathcal{G}

A τ instance \mathcal{I} is called a **solution** for \mathcal{G} under \mathcal{M} iff

$(\mathcal{G}, \mathcal{I})$ satisfies all rules in $M_{\sigma\tau}$ (for short: $(\mathcal{G}, \mathcal{I}) \models M_{\sigma\tau}$) and \mathcal{I} satisfies all rules in M_{τ} .

Reminder: Solutions

Definition

Given: a mapping \mathcal{M} and a σ instance \mathcal{G}

A τ instance \mathcal{I} is called a **solution** for \mathcal{G} under \mathcal{M} iff

$(\mathcal{G}, \mathcal{I})$ satisfies all rules in $M_{\sigma\tau}$ (for short: $(\mathcal{G}, \mathcal{I}) \models M_{\sigma\tau}$) and \mathcal{I} satisfies all rules in M_{τ} .

- ▶ $(\mathcal{G}, \mathcal{I}) \models M_{\sigma\tau}$ iff $\mathcal{G} \cup \mathcal{I} \models M_{\sigma\tau}$ where
 - ▶ $\mathcal{G} \cup \mathcal{I}$ is the union of the instances \mathcal{G}, \mathcal{I} : Structure containing all relations from \mathcal{G} and \mathcal{I} with domain the union of domains of \mathcal{G} and \mathcal{I}
 - ▶ well defined because schemata are disjoint
- ▶ $Sol_{\mathcal{M}}(\mathcal{G})$: Set of solutions for \mathcal{G} under \mathcal{M}

First Key Problem: Existence of Solutions

Problem: SOLEXISTENCE $_{\mathcal{M}}$

Input: Source instance \mathcal{G}

Output: Answer whether there exists a solution for \mathcal{G} under \mathcal{M}

- ▶ Note: \mathcal{M} is assumed to be fixed \implies data complexity
- ▶ This problem is going to be approached with a well known proof tool: *chase*

Trivial Case: No Target Dependencies

- ▶ Without target constraints there is always a solution

Proposition

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ with $M_{\sigma\tau}$ consisting of st-tgds. Then for any source instance \mathfrak{S} there are infinitely many solutions and at least one solution can be constructed in polynomial time.

Trivial Case: No Target Dependencies

- ▶ Without target constraints there is always a solution

Proposition

Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ with $M_{\sigma\tau}$ consisting of st-tgds. Then for any source instance \mathfrak{S} there are infinitely many solutions and at least one solution can be constructed in polynomial time.

Proof Idea

- ▶ For every rule and every tuple \vec{a} fulfilling the antecedens generate facts according to the succedens (using fresh named nulls for the existentially quantified variables)
- ▶ Resulting τ instance \mathfrak{T} is a solution
- ▶ Polynomial: Testing whether \vec{a} fulfills the head (a conjunctive query) can be done in polynomial time
- ▶ Infinity: From \mathfrak{T} can build any other solution by extension

Undecidability for General Constraints

Theorem

There is a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_{\tau})$ such that $SOEXISTENCE_{\mathcal{M}}$ is undecidable.

- ▶ Proof by reduction from embedding problem for finite semigroups which is known to be undecidable (Arenas et al. 2014, Thm 5.3)
- ▶ As a consequence: Further restrict mapping rules
- ▶ But note that the following chase construction defined for arbitrary st-tgds

Chase Construction

- ▶ A widely used tool in DB theory
- ▶ Original use: Calculating entailments of DB constraints (Maier et al, 1979)
- ▶ **Idea**
 - ▶ Apply tgds as completion/repair rules in a bottom-up strategy
 - ▶ until no tgds can be applied anymore
 - ▶ Chase construction may fail if one of the egds is violated
- ▶ The chase leads to an instance with desirable properties
 - ▶ It produces not too many redundant facts
 - ▶ Universality

Lit: D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, Dec. 1979.

Example (Terminating chase)

- ▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

- ▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$$M_{\tau} = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1} \}$$

- ▶ Source instance $\mathcal{G} = \{E(a, b)\}$
- ▶ Going to build stepwise potential target instances \mathcal{T}_i considering pairs $(\mathcal{G}, \mathcal{T}_i)$

Example (Terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ Going to build stepwise potential target instances \mathfrak{T}_i considering pairs $(\mathfrak{G}, \mathfrak{T}_i)$

▶ $(\mathfrak{G}, \emptyset)$

Example (Terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ Going to build stepwise potential target instances \mathfrak{T}_i
considering pairs $(\mathfrak{G}, \mathfrak{T}_i)$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

Example (Terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ Going to build stepwise potential target instances \mathfrak{T}_i
considering pairs $(\mathfrak{G}, \mathfrak{T}_i)$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$

Example (Terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ Going to build stepwise potential target instances \mathfrak{T}_i considering pairs $(\mathfrak{G}, \mathfrak{T}_i)$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp)\})$ (termination)

Example (Terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ Going to build stepwise potential target instances \mathfrak{T}_i considering pairs $(\mathfrak{G}, \mathfrak{T}_i)$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp)\})$ (termination)

Example (Non-terminating chase)

- ▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- ▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$
- $M_{\tau} = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$
- ▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

Example (Non-terminating chase)

- ▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- ▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$
- ▶ $M_{\tau} = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$
- ▶ Source instance $\mathfrak{G} = \{E(a, b)\}$
- ▶ $(\mathfrak{G}, \emptyset)$

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ $(\mathfrak{G}, \emptyset)$

(violates θ_1)

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{S} = \{E(a, b)\}$

▶ $(\mathfrak{S}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{S}, \{G(a, b)\})$

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp)\})$

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp)\})$ (violates χ_2)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp), G(\perp, \perp_1)\})$

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp), G(\perp, \perp_1), L(\perp_1, \perp_2)\})$

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp)\})$ (violates χ_2)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp), G(\perp, \perp_1)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp), G(\perp, \perp_1), L(\perp_1, \perp_2)\})$

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{S} = \{E(a, b)\}$

▶ $(\mathfrak{S}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{S}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{S}, \{G(a, b), L(b, \perp)\})$ (violates χ_2)

▶ $(\mathfrak{S}, \{G(a, b), L(b, \perp), G(\perp, \perp_1)\})$ (violates χ_1)

▶ $(\mathfrak{S}, \{G(a, b), L(b, \perp), G(\perp, \perp_1), L(\perp_1, \perp_2)\})$ (violates χ_2)

Example (Non-terminating chase)

▶ Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$

▶ $M_{\sigma\tau} = \{ \underbrace{E(x, y) \rightarrow G(x, y)}_{\theta_1} \}$

$M_\tau = \{ \underbrace{G(x, y) \rightarrow \exists z L(y, z)}_{\chi_1}, \underbrace{L(x, y) \rightarrow \exists z G(y, z)}_{\chi_2} \}$

▶ Source instance $\mathfrak{G} = \{E(a, b)\}$

▶ $(\mathfrak{G}, \emptyset)$ (violates θ_1)

▶ $(\mathfrak{G}, \{G(a, b)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp)\})$ (violates χ_2)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp), G(\perp, \perp_1)\})$ (violates χ_1)

▶ $(\mathfrak{G}, \{G(a, b), L(b, \perp), G(\perp, \perp_1), L(\perp_1, \perp_2)\})$ (violates χ_2)

▶ ... (non-termination)

Chase Definition

- ▶ Let \mathcal{G} be a σ instance and $\text{dom}(\mathcal{G})$ its domain

Definition (Chase steps)

$$\boxed{\mathcal{G} \xrightarrow{\chi, \vec{a}} \mathcal{G}'} \text{ iff}$$

Chase Definition

- ▶ Let \mathcal{G} be a σ instance and $dom(\mathcal{G})$ its domain

Definition (Chase steps)

$\mathcal{G} \xrightarrow{\chi, \vec{a}} \mathcal{G}'$ iff

1. χ is a **tgd** of the form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
 - ▶ $\mathcal{G} \models \phi(\vec{a})$ for some elements \vec{a} from $dom(\mathcal{G})$
 - ▶ \mathcal{G}' extends \mathcal{G} with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.

Chase Definition

- ▶ Let \mathcal{G} be a σ instance and $dom(\mathcal{G})$ its domain

Definition (Chase steps)

$\mathcal{G} \xrightarrow{\chi, \vec{a}} \mathcal{G}'$ iff

1. χ is a **tgd** of the form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
 - ▶ $\mathcal{G} \models \phi(\vec{a})$ for some elements \vec{a} from $dom(\mathcal{G})$
 - ▶ \mathcal{G}' extends \mathcal{G} with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.
2. or χ is an **egd** of form $\phi(\vec{x}) \rightarrow x_i = x_j$ and
 - ▶ $\mathcal{G} \models \phi(\vec{a})$ for some elements \vec{a} from $dom(\mathcal{G})$ with $a_i \neq a_j$ and
 - ▶ (a_i is constant or a null, a_j is a null and $\mathcal{G}' = \mathcal{G}[a_j/a_i]$ or a_i is a null, a_j is constant and $\mathcal{G}' = \mathcal{G}[a_i/a_j]$)

Chase Definition

- ▶ Let \mathcal{G} be a σ instance and $dom(\mathcal{G})$ its domain

Definition (Chase steps)

$\mathcal{G} \xrightarrow{\chi, \vec{a}} \mathcal{G}'$ iff

1. χ is a **tg**d of the form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
 - ▶ $\mathcal{G} \models \phi(\vec{a})$ for some elements \vec{a} from $dom(\mathcal{G})$
 - ▶ \mathcal{G}' extends \mathcal{G} with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.
2. or χ is an **egd** of form $\phi(\vec{x}) \rightarrow x_i = x_j$ and
 - ▶ $\mathcal{G} \models \phi(\vec{a})$ for some elements \vec{a} from $dom(\mathcal{G})$ with $a_i \neq a_j$ and
 - ▶ (a_i is constant or a null, a_j is a null and $\mathcal{G}' = \mathcal{G}[a_j/a_i]$ or a_i is a null, a_j is constant and $\mathcal{G}' = \mathcal{G}[a_i/a_j]$)

$\mathcal{G} \xrightarrow{\chi, \vec{a}} fail$ iff

- ▶ $\mathcal{G} \models \phi(\vec{a})$ for some elements \vec{a} from $dom(\mathcal{G})$ with $a_i \neq a_j$
- ▶ and both a_i, a_j are constants.

Chase

Definition

A chase sequence for \mathcal{G} under M is a sequence of chase steps

$\mathcal{G}_i \xrightarrow{\chi_i, \vec{a}_i} \mathcal{G}_{i+1}$ such that

- ▶ $\mathcal{G}_0 = \mathcal{G}$
- ▶ each χ_i is in M
- ▶ for each distinct i, j also $(\chi_i, \vec{a}_i) \neq (\chi_j, \vec{a}_j)$

For a finite chase sequence the last instance is called its **result**.

- ▶ If the result is **fail**, then the sequence is said to be a **failing sequence**
- ▶ If no further dependency from M can be applied to a result, then the sequence is called **successful**.

Indeterminism in Chase Construction

- ▶ Indeterminism regarding choice of nulls (no problem)
- ▶ Indeterminism regarding order of chosen tgds and egds
This may lead to different chase results.

Use of Chases in Data Exchange

- ▶ A chase sequence for \mathcal{G} under a \mathcal{M} is a chase sequence for (\mathcal{G}, \emptyset) under $M_{\sigma\tau} \cup M_{\tau}$
- ▶ If $(\mathcal{G}, \mathcal{I})$ result of a finite sequence, call just \mathcal{I} the result
- ▶ Chase is the right tool for finding solutions

Proposition

Given \mathcal{M} and source instance \mathcal{G} .

- ▶ *If there is a successful chase sequence for \mathcal{G} with result \mathcal{I} , then \mathcal{I} is a solution.*
- ▶ *If there is a failing chase sequence for \mathcal{G} , then \mathcal{G} has no solution.*

Use of Chases in Data Exchange

- ▶ A chase sequence for \mathcal{G} under a \mathcal{M} is a chase sequence for (\mathcal{G}, \emptyset) under $M_{\sigma\tau} \cup M_{\tau}$
- ▶ If $(\mathcal{G}, \mathcal{T})$ result of a finite sequence, call just \mathcal{T} the result
- ▶ Chase is the right tool for finding solutions

Proposition

Given \mathcal{M} and source instance \mathcal{G} .

- ▶ *If there is a successful chase sequence for \mathcal{G} with result \mathcal{T} , then \mathcal{T} is a solution.*
 - ▶ *If there is a failing chase sequence for \mathcal{G} , then \mathcal{G} has no solution.*
- ▶ The proposition does not cover all cases: non-terminating chase
 - ▶ In this case there may still be a solution

Weak Acyclicity

- ▶ In order to guarantee termination, restrict target constraints
- ▶ Reason for non-termination: generation of new nulls with same dependencies

Example (Cycle in Dependencies)

- ▶ $\chi_1 = G(x, y) \rightarrow \exists z L(y, z)$
- ▶ $\chi_2 = L(x, y) \rightarrow \exists z G(y, z)$

Weak Acyclicity

- ▶ In order to guarantee termination, restrict target constraints
- ▶ Reason for non-termination: generation of new nulls with same dependencies

Example (Cycle in Dependencies)

- ▶ $\chi_1 = G(x, y) \rightarrow \exists z L(y, z)$
- ▶ $\chi_2 = L(x, y) \rightarrow \exists z G(y, z)$

Possible infinite generation

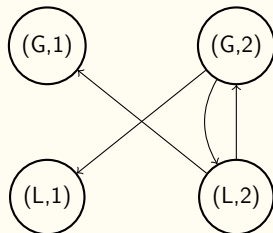
$$G(a, b) \xrightarrow{\chi_1} L(b, \perp_1) \xrightarrow{\chi_2} G(\perp_1, \perp_2) \xrightarrow{\chi_1} L(\perp_2, \perp_3) \dots$$

Simple Dependency Graphs

- ▶ Nodes: pairs (R, i) of predicate R and argument-position i
- ▶ Edges: From (R_b, i) to (R_h, j) iff there is a tgd $\forall \vec{x} \forall \vec{y} \phi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \psi(\vec{x}, \vec{z})$ and
 1. R_h occurs in ψ and R_b occurs in ϕ and
 2. for $x \in \vec{x}$ in i -position in R_b
 - ▶ either x occurs in j -position in R_h
 - ▶ or the variable in j -position in R_h is existentially quantified

Example (Simple Dependency Graph with Cycle)

- ▶ $\chi_1 = G(y, x) \rightarrow \exists z L(x, z)$
- ▶ $\chi_2 = L(y, x) \rightarrow \exists z G(x, z)$



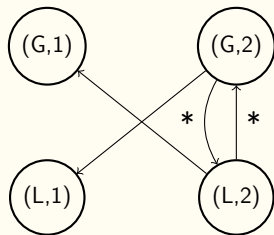
Set of tgds called **acyclic** if simple dependency graph is acyclic.

Dependency Graphs (DG)

- ▶ Nodes: pairs (R, i) of predicate R and argument-position i
- ▶ Edges: From (R_b, i) to (R_h, j) iff there is a tgd $\forall \vec{x} \forall \vec{y} \phi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \psi(\vec{x}, \vec{z})$ and
 1. R_h occurs in ψ and R_b occurs in ϕ and
 2. for $x \in \vec{x}$ in i -position in R_b
 - ▶ either x occurs in j -position in R_h
 - ▶ or the variable in j -position in R_h is existentially quantified; in this case **the edge is labelled by ***

Example (Not weakly acyclic Dependency Graph)

- ▶ $\chi_1 = G(y, x) \rightarrow \exists z L(x, z)$
- ▶ $\chi_2 = L(y, x) \rightarrow \exists z G(x, z)$



TGDs **weakly** acyclic iff DG has no cycle with a * edge.

Termination for weakly acyclic tgds

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 the length of every chase sequence for a source \mathcal{G} is polynomially bounded w.r.t. the size of \mathcal{G} .

- ▶ In particular: Every chase sequence terminates
- ▶ Moreover: $\text{SOLEXISTENCE}_{\mathcal{M}}$ can be solved in polynomial time
- ▶ a solution can be constructed in polynomial time

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 reduction of halting problem to this problem

Theorem

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

Desiderata

- ▶ Due to the undecidability 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 exist, 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 $\mathfrak{T}_1 \xrightarrow{\chi} \mathfrak{T}_2$ by dependency χ , \mathfrak{T}_3 fulfills χ and there is $h : \mathfrak{T}_1 \xrightarrow{hom} \mathfrak{T}_3$, then there is $h' : \mathfrak{T}_2 \xrightarrow{hom} \mathfrak{T}_3$)
- ▶ Argue inductively starting from empty database \emptyset and identity homomorphism $\emptyset \xrightarrow{id} \mathfrak{T}'$.

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).

Main Theorem for Cores

Theorem

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