Information Integration

Part 3: Information Integration Models

Werner Nutt

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2012/2013

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Information Integration

II has the aim to provide **uniform access** to data that are stored in **a number** of **autonomous and heterogeneous** sources:

- different *data models* (structured, semi-structured, text)
- different *schemata*
- differences in the representation of *values* (km vs. miles, USD vs. EUR) and *entities* (addresses, dates, etc.)
- *inconsistencies* among the data

II is a **basic problem** in

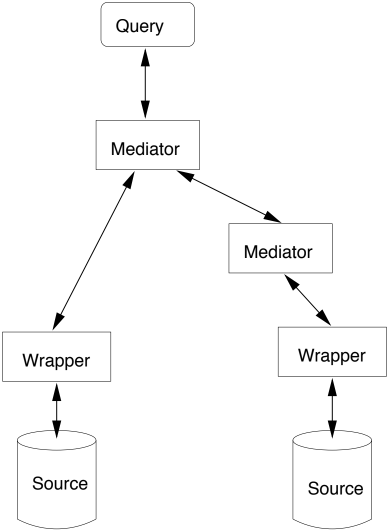
- Data Warehousing, Data Re-engineering
- Integration of data from scientific experiments
- E-commerce: Harvesting data on the Web

Architecture of a Mediator-based II System

The system generates an *integrated, uniform view* of a *collection of sources*

- Queries are formulated over a **global schema**
  domain model, domain schema, "mediated schema",
  ontology, enterprise model, . . .
- **Wrappers** *(= cover, envelop, encase)*
  make sources *accessible*
- **Mediators** *translate* queries,
  combine answers of wrappers and mediators,
  resolve contradictions

Information Integration: Scenario



Movie Info: Global Schema [Idea by A. Halevy]

Movie(title, director, year, genre, rating)

Starring(title, actor)

Artist(name, yob, country)

Plays(title, language, cinema, startTime)

Cinema(cinema, location)

Review(title, rating, description)

Movie Info: Queries Over the Global Schema

- *"Which films with Johnny Depp are shown in Bolzano at which time?"*

$$Q(t, st) \text{ :– } \texttt{Starring}(t, \text{'Johnny Depp'}), \texttt{Plays}(t, l, c, st), \texttt{Cinema}(c, \text{'Bolzano'})$$

- *"Which thrillers by an Italian director are shown in Bolzano at which time?"*

$$Q(t, st) \text{ :– } \texttt{Movie}(t, d, y, \text{'Thriller'}), \texttt{Artist}(d, \text{'Italy'}), \texttt{Plays}(t, l, c, st), \texttt{Cinema}(c, \text{'Bolzano'})$$

Movie Info: Sources

- Website Cineplexx Cinema, Bozen

  `CineplexxShowing(title, language, startTime)`
  `CineplexxDetails(title, director, genre)`
  `CineplexxCast(title, actor)`

- Website Filmclub Cinema, Bozen

  `Filmclub(title, language, director, startTime)`

- Website Kinoliste

  `Kinoliste(city, cinema)`

- Internet Movie Database

  `ImdbActor(name, yob)`
  `ImdbStarring(name, title)`
  `ImdbFilm1(title, stars, genre, director, year)`
  `ImdbFilm2(title, actor)`
  `ImdbReview(title, stars, description)`

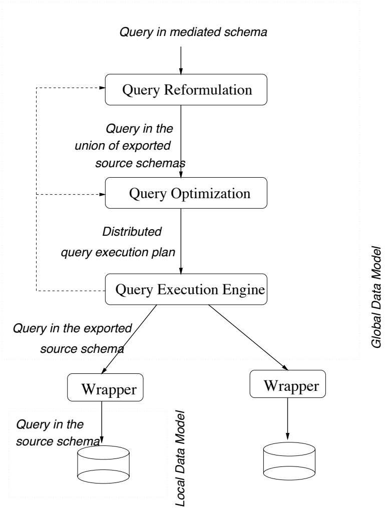
- Website Kino München

  `KinoMuenchen(cinema, title, startTime)`

Approaches to II

- What does the II system contain?
  $\Rightarrow$ **virtual** vs. **materialized integration**
- Which *operations* are allowed on the global schema?
  $\Rightarrow$ **Read** vs. Read and Write
- How is the II system *specified*?
  $\Rightarrow$ procedurally vs. **declaratively**
- How do we model the *connection* between sources and global schema?
  $\Rightarrow$ global schema **in terms of the sources**
  vs. sources **in terms of the global schema**

Architecture of a Virtual Integration System



Questions about II [M. Lenzerini]

- How to construct the global schema
- (Automatic) source wrapping
- How to express mappings between sources and global schema
- How to discover mappings between sources and global schema
- How to deal with limitations in mechanisms for accessing sources
- Data extraction, cleaning, and reconciliation
- How to model the global schema, the sources, and the mappings
- How to answer queries expressed on the global schema
- How to exchange data according to the mappings
- How to optimize query answering
- How to process updates expressed on the global schema and/or the sources (read/write vs. read-only data integration)

Questions about II [M. Lenzerini]

- How to construct the global schema
- (Automatic) source wrapping
- How to express mappings between sources and global schema
- How to discover mappings between sources and global schema
- How to deal with limitations in mechanisms for accessing sources
- Data extraction, cleaning, and reconciliation
- How to model the global schema, the sources, and the mappings
- How to answer queries expressed on the global schema
- How to exchange data according to the mappings
- How to optimize query answering
- How to process updates expressed on the global schema and/or the sources (read/write vs. read-only data integration)

The Mediator (1)

The mediator

- provides an *integrated access* to the information sources
- *hides the sources*
- creates the illusion to query a unique database

$\leadsto$ the mediator presents to the user a *virtual db*

$\leadsto$ the virtual db is presented by a *schema:*
the *global* or *mediated schema*

Depending on the application, several data models are possible:
relational, XML, description logics

Here: the global schema is a **relational schema**

The Mediator (2)

Function

- accepts a *query* over the *global schema*
- *reformulates the query* into queries over the sources
- determines an *execution plan*: in which *order* will the queries be posed over the sources?

  (information flow size of the expected answers,
  expected speed of the answer)
- *sends* queries to the sources (= wrappers)
- *collects* and *combines* the answers
- *changes* the plan during run time

Modeling the Information Content of Sources

2 approaches of mapping source schemas and global schema

- Relations in the *global schema* are *views of the sources:*
  "global as view" (GAV)

  traditional concept of a view
  - Views are virtual relations
    the global schema describes a virtual DB

- Relations in the *sources* are *views of the global schema:*
  "local as view" (LAV)

  apparently nonsensical
  - sources are materialized views of a db,
    which is not accessible itself

There is also a combination of the two, called GLAV

Logical Query Planning

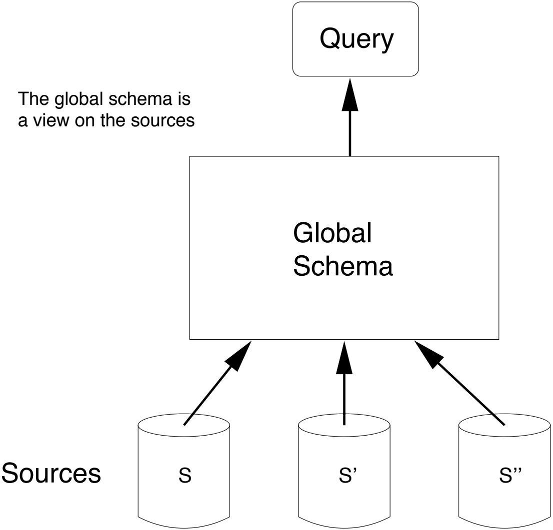
In a standard database setting (centralized or distributed):

- Given: a declarative query over the logical schema
- Wanted: a sequence of operations for retrieving data, operating on the physical schema:
  the **execution** plan

In information integration:

- Given: a declarative query over the global schema
- Wanted: an "equivalent" declarative query over the local schemas:
  the **logical** plan
- The logical plan can be transformed into an execution plan with (more or less) standard techniques

Global as View: Idea



Global as View: Example (1) [J. Ullman]

Sources: $\mathtt{S}_1$, $\mathtt{S}_2$, $\mathtt{S}_3$ contain info on *employees* e, *phone numbers* p, *managers* m, *offices* o, *departments* d. Thus, the source schema is:

$$\mathtt{S}_1(e,p,m) \qquad \mathtt{S}_2(e,o,d) \qquad \mathtt{S}_3(e,p),$$

where variable names indicate the meaning of the positions.

Global Schema: We combine the three sources into a global schema with the two relations EPO and EDM:

$$\begin{aligned} \mathtt{EPO}(e,p,o) &:\!-\ \mathtt{S}_1(e,p,m),\ \mathtt{S}_2(e,o,d) \\ \mathtt{EPO}(e,p,o) &:\!-\ \mathtt{S}_3(e,p),\ \mathtt{S}_2(e,o,d) \\ \mathtt{EDM}(e,d,m) &:\!-\ \mathtt{S}_1(e,p,m),\ \mathtt{S}_2(e,o,d) \end{aligned}$$

EPO und EDM are described by views on the sources

Global as View: Example (2)

Query 1: "What are Sally's phone and office?"

$$Q_1(p, o) \coloneq \mathtt{EPO}(\textit{'Sally'}, p, o)$$

We obtain a plan P_1 for Q_1 if we expand the body of Q_1, by unfolding the predicate EPO:

$$P_1(p, o) \coloneq \mathtt{S}_1(\textit{'Sally'}, p, m),\ \mathtt{S}_2(\textit{'Sally'}, o, d)$$
$$P_1(p, o) \coloneq \mathtt{S}_3(\textit{'Sally'}, p),\ \mathtt{S}_2(\textit{'Sally'}, o, d)$$

Global as View: Example (3)

Query 2: "What are Sally's office and department?"

$$Q_2(o,d) \coloneq \texttt{EPO}(\textit{'Sally'}, p, o),\ \texttt{EDM}(\textit{'Sally'}, d, m)$$

Again, if we expand the body of Q_2 unfolding the definitions of EPO and EDM, we obtain a plan P_2 for Q_2:

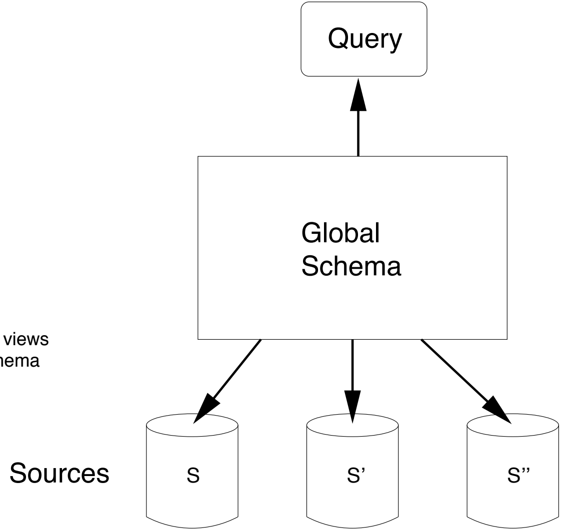
$$\begin{aligned} P_2(o,d) \coloneq\ & \texttt{S}_1(\textit{'Sally'}, p1, m1),\ \texttt{S}_2(\textit{'Sally'}, o, d), \\ & \texttt{S}_1(\textit{'Sally'}, p2, m2),\ \texttt{S}_2(\textit{'Sally'}, o, d) \end{aligned}$$

$$\begin{aligned} P_2(o,d) \coloneq\ & \texttt{S}_3(\textit{'Sally'}, p1),\ \texttt{S}_2(\textit{'Sally'}, o, d), \\ & \texttt{S}_1(\textit{'Sally'}, p2, m),\ \texttt{S}_2(\textit{'Sally'}, o, d) \end{aligned}$$

But: Wouldn't a single plan be sufficient

$$P'_2(o,d) \quad \coloneq \quad S_2(\textit{'Sally'}, o, d)?$$

Local as View: Idea



The sources are views on the global schema

Local as View: Example (1)

Sources: Again, we have the same three sources $\texttt{S}_1$, $\texttt{S}_2$, $\texttt{S}_3$:

$$\texttt{S}_1(e, p, m) \qquad \texttt{S}_2(e, o, d) \qquad \texttt{S}_3(e, p)$$

Global Schema: We model the application domain by five relations:

- $\texttt{Emp}(e)$: e is an employee
- $\texttt{Phone}(e, p)$: e has phone number p
- $\texttt{Office}(e, o)$: e has office o
- $\texttt{Mgr}(e, m)$: m is e's manager
- $\texttt{Dept}(e, d)$: d is e's department

Local as View: Example (2)

Source Descriptions: We describe the sources as being included in views on the global schema:

$$\mathtt{S}_1 \subseteq V_1 \qquad \mathtt{S}_2 \subseteq V_2 \qquad \mathtt{S}_3 \subseteq V_3.$$

The views have the following definitions:

$$\begin{aligned} V_1(e,p,m) &\mathrel{:-} \mathtt{Emp}(e),\ \mathtt{Phone}(e,p),\ \mathtt{Mgr}(e,m) \\ V_2(e,o,d) &\mathrel{:-} \mathtt{Emp}(e),\ \mathtt{Office}(e,o),\ \mathtt{Dept}(e,d) \\ V_3(e,p) &\mathrel{:-} \mathtt{Emp}(e),\ \mathtt{Phone}(e,p) \end{aligned}$$

Local as View: Example (3)

Query 3: "What are Sally's phone and office?"

$$Q_3(p, o) \ :- \ \texttt{Phone}(\textit{'Sally'}, p), \texttt{Office}(\textit{'Sally'}, p)$$

Problem: No source contains complete information about phone numbers and offices. Moreover, the information we are looking for is always combined with other information.

Idea: Use the views to construct queries that are equivalent or more specific than Q_3:

$$P_3(p, o) :- V_1(\textit{'Sally'}, p, m), V_2(\textit{'Sally'}, o, d)$$
$$P_3(p, o) :- V_3(\textit{'Sally'}, p), V_2(\textit{'Sally'}, o, d).$$

How can we test that P_3 is equivalent to or more specfic than Q_3?

$\leadsto$ *Unfold the views!*

Local as View: Example (4)

Unfolding: We use the superscript $\cdot^{unf}$ to indicate unfolding using definitions:

$$P_3^{unf}(p,o) \coloneq \texttt{Emp}(\textit{'Sally'}), \texttt{Phone}(\textit{'Sally'}, p), \texttt{Mgr}(\textit{'Sally'}, d),$$
$$\texttt{Emp}(\textit{'Sally'}), \texttt{Office}(\textit{'Sally'}, o), \texttt{Dept}(\textit{'Sally'}, d)$$

$$P_3^{unf}(p,o) \coloneq \texttt{Emp}(\textit{'Sally'}), \texttt{Phone}(\textit{'Sally'}, p),$$
$$\texttt{Emp}(\textit{'Sally'}), \texttt{Office}(\textit{'Sally'}, o), \texttt{Dept}(\textit{'Sally'}, d)$$

Each rule of P_3^{unf} has "more" (in the sense of "$\supseteq$") conditions than Q_3:

$\Rightarrow$ Q_3 contains each rule of P_3^{unf}

$\Rightarrow$ Q_3 contains P_3^{unf}

Local as View: Example (5)

Query 4: "What are Sally's office and department?"

$$Q_4(o, d) \text{ :- } \texttt{Office}(\textit{'Sally'}, o), \texttt{Dept}(\textit{'Sally'}, d)$$

Office and departments are only mentioned in V_2. Hence:

$$P_4(o, d) \text{ :- } V_2(\textit{'Sally'}, o, d)$$

Unfolding:

$$P_4^{\mathit{unf}}(o, d) \text{ :- } \texttt{Emp}(\textit{'Sally'}), \texttt{Office}(\textit{'Sally'}, o), \texttt{Dept}(\textit{'Sally'}, d)$$

Again, the plan is contained in the query, thus okay ...

Global as View vs. Local as View

Global as View:

+ *query reformulation* is **simple**: unfold *(. . . and simplify!)*
+ **abstracts** from *irrelevant information* in the sources
  (e.g., can forget attributes)
− **changes** in the *sources* **affect** the *global schema*
− **connections** between the *sources*
  need to be taken into account **when setting up the schema**
  ("query reformulation at design time")

Local as View:

+ **modularity** and **reusability**:
  when a source changes, only its description needs to be changed
+ **connections** between the sources can be **inferred**
− **query processing** is difficult: "query reformulation at run time"

Questions

We started with queries Q over the global schema
and transformed them to queries Q' over the sources

Are these transformations

- **correct?**
  that is, are all answers to Q' also answers to Q?
- **complete?**
  that is, will Q' retrieve all (sensible) answers for Q?
- generally **computable?**

$\rightsquigarrow$ What at all are answers for Q?

Information Integration Systems

Here: formal framework for

- defining the problems of II (= information integration)
- developing and comparing techniques
- comparing approaches

Ideas:

- sources are accessed by means of a **global schema** $\mathcal{G}$, which describes a virtual db
- the instance $\mathbf{J}$ of the virtual db is *unknown*
- the source instance $\mathbf{I}$ *restricts* the possible global instances $\mathbf{J}$
  $\leadsto$ how can one model the connection between sources and virtual db?

$\leadsto$ Queries over $\mathcal{G}$ must be answered with **incomplete information**

Incomplete Information

Schema

`Person(fname, surname, city, street)`

`City(cname, population)`

We know

- Mair lives in Bozen *(but we don't know first name and street)*
- Carlo Rossi lives in Bozen *(but we don't know the street)*
- Mair and Carlo Rossi live in the same street *(but we don't know which)*
- Maria Pichler lives in Brixen
- Bozen has a population of 100,500
- Brixen has a population $<$ 100,000 *(but we don't known the number)*

Queries

1. Return first name and surname of people living in Bozen!
2. Return the surnames of people living in Bozen!
3. Who (surname) is living in the same street as Mair?
4. Which people are living in a city with less than 100,000 inhabitants?

Incomplete Information: Questions

How can we represent this info?

- What does the representation look like?
- What is its meaning?
- What answers should the queries return?
- How can we define the semantics of queries?

Modeling Incomplete Information: SQL Nulls

Person

fname	surname	city	street
NULL	Mair	Bozen	NULL
Carlo	Rossi	Bozen	NULL
Maria	Pichler	Brixen	NULL

City

cname	population
Bozen	100,500
Brixen	NULL

Intuitive meaning of NULL: one of

(i) does not exist
(ii) exists, but is unknown
(iii) unknown whether (i) or (ii)

SQL Nulls: Formal Semantics

- **dom** (or, equivalenty, every type) is extended by a new value: `NULL`
- built-in predicates are evaluated according to a 3-valued logic with truth values *false* $<$ *unknown* $<$ *true*
- atoms with `NULL` evaluate to *unknown*
- Boolean operations:
  - AND/OR correspond to min/max on truth values
  - NOT extends the classical definition by NOT(*unknown*) $=$ *unknown*
- additional operation $\mathtt{ISNULL}(\cdot)$ with $\mathtt{ISNULL}(v) =$ *true* iff v is `NULL`
- a query returns those tuples for which query conditions evaluate to *true*

SQL Nulls: Example Queries

- Query 1 returns (`NULL`, Mair) and (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Queries 3 and 4 return nothing

Representation Systems [Imieliński/Lipski, 1984]

Distinguish between

- **semantic instances**, which are the ones we know
- **syntactic instances**, which contain tuples with variables (written $\perp_1$, $\perp_2, \ldots$)

A syntactic instance represents many semantic instances

Syntactic instances are called **multi-tables** (i.e., several tables).

There are three kinds of (multi-)tables:

Codd Tables: a variable occurs no more than once

Naive or Variable Tables: a variable can occur several times

Conditional Tables: variable table where each tuple $\bar{t}$ is tagged with a boolean combination *cond*$(\bar{t})$ of built-in atoms

Short names: table, v-table, c-table

Semantics of Tables

Let $\mathbf{T}$ be a multi-table with variables $\mathit{var}(\mathbf{T})$.
For an assignement $\alpha\colon \mathit{var}(\mathbf{T}) \to \mathbf{dom}$ we define

$$\alpha\mathbf{T} = \{\alpha\,\bar{t} \mid \bar{t} \in \mathbf{T},\ \alpha \models \mathit{cond}(\bar{t}\,)\}$$

Then $\mathbf{T}$ **represents** the infinite sets of instances

$$\mathit{rep}(\mathbf{T}) = \{\alpha\mathbf{T} \mid \alpha\colon \mathit{var}(\mathbf{T}) \to \mathbf{dom}\}$$

$$\mathit{Rep}(\mathbf{T}) = \{\mathbf{J} \mid \mathbf{I} \subseteq \mathbf{J} \text{ for some } \mathbf{I} \in \mathit{rep}(\mathbf{T})\}$$

where

$\mathit{rep}(\mathbf{T})$ is the *closed-world* interpretation of $\mathbf{T}$

$\mathit{Rep}(\mathbf{T})$ is the *open-world* interpretation of $\mathbf{T}$

(Many results hold for both, the closed-world and the open-world interpretation. We assume open-world interpretation if not said otherwise.)

Certain and Possible Answers

Given **T** and a query Q, the tuple $\bar{c}$ is

- a **certain answer** (for Q over **T**) if
  $\bar{c}$ is returned by Q over **all** instances represented by **T**
- a **possible answer** if
  $\bar{c}$ is returned by Q over **some** instance represented by **T**

We denote the set of all certain answers as $\mathit{cert}_{\mathbf{T}}(Q)$.

We have

$$\mathit{cert}_{\mathbf{T}}(Q) \quad = \bigcap_{\mathbf{J} \in \mathit{Rep}(\mathbf{T})} Q(\mathbf{J})$$

Modeling Incomplete Information: Codd-Tables

`Person`

fname	surname	city	street
$\perp_1$	Mair	Bozen	$\perp_2$
Carlo	Rossi	Bozen	$\perp_3$
Maria	Pichler	Brixen	$\perp_4$

`City`

cname	population
Bozen	100,500
Brixen	$\perp_5$

Certain answers for our example queries:

- Query 1 returns (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Query 3 returns **Mair**
- Query 4 returns nothing

Modeling Incomplete Information: v-Tables

Person

fname	surname	city	street
$\bot_1$	Mair	Bozen	$\bot_2$
Carlo	Rossi	Bozen	$\bot_2$
Maria	Pichler	Brixen	$\bot_4$

City

cname	population
Bozen	100,500
Brixen	$\bot_5$

Certain answers for our example queries:

- Query 1 returns (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Query 3 returns Mair and **Rossi**
- Query 4 returns nothing

Modeling Incomplete Information: c-Tables

Person

fname	surname	city	street
$\perp_1$	Mair	Bozen	$\perp_2$
Carlo	Rossi	Bozen	$\perp_2$
Maria	Pichler	Brixen	$\perp_4$

City

cname	population	cond
Bozen	100,500	*true*
Brixen	$\perp_5$	$\perp_5 < 100,000$

Certain answers for our example queries:

- Query 1 returns (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Query 3 returns Mair and Rossi
- Query 4 returns **Pichler**

Strong Representation Systems

Definition

Let Q be a query and $\mathbf{T}$ be a table. Then

$$Q(\mathbf{T}) := \{Q(\mathbf{I}) \mid \mathbf{I} \in \mathit{rep}(\mathbf{T})\}$$

That is, $Q(\mathbf{T})$ contains the relation instances obtained by applying Q individually to each instance represented by $\mathbf{T}$.

Note: $Q(\mathbf{T})$ is a set of sets of tuples, not a set of tuples!

Strong Representation Systems (cont)

Theorem (Imieliński/Lipski)

For every relational algebra query Q and every c-table $\mathbf{T}$ one can compute a c-table $\widetilde{\mathbf{T}}$ such that

$$rep(\widetilde{\mathbf{T}}) = Q(\mathbf{T})$$

That is,

- $\widetilde{\mathbf{T}}$ can be considered the answer of Q over $\mathbf{T}$
- the result of querying a c-table can be represented by a c-table
  $\rightsquigarrow$ c-tables are a **strong representation system**

The downside:

- handling of c-tables is intractable:
  the membership problem "$\mathbf{I} \in rep(\mathbf{T})$"? is NP-hard
- the c-tables $\widetilde{\mathbf{T}}$ may be very large

Weak Representation Systems: Motivation

Let $\mathbf{T}_{\mathrm{v}}$ be our example v-table and consider

$$Q_0 = \pi_{\mathtt{fname},\mathtt{sname}}(\sigma_{\mathtt{city}=\text{'Bozen'}}(\mathtt{Person})),$$
$$Q_1 = \pi_{\mathtt{sname}}(\sigma_{\mathtt{city}=\text{'Bozen'}}(\mathtt{Person}))$$

Then: $\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_0) = \{(\text{Carlo, Rossi})\}$ and
$\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_1) = \{(\text{Mair}), (\text{Rossi})\}$

Observation: $Q_0 = \pi_{\mathtt{sname}}(Q_1)$,
but $\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_0)$ cannot be computed from $\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_1)$

Compositionality is violated! Better keep the nulls!

Idea: Try to keep enough information for representing certain answers!

Incomplete Databases: Definition

Definition (Incomplete Database)

An **incomplete database** is a set of instances ($\mathcal{I}$, $\mathcal{J}$).

For a query Q and an incomplete db $\mathcal{I}$, the set of certain answers for Q over $\mathcal{I}$ is

$$\mathit{cert}_{\mathcal{I}}(Q) := \bigcap_{\mathbf{I} \in \mathcal{I}} Q(\mathbf{I}).$$

Weak Representation Systems

Let $\mathcal{L}$ be a query language
(e.g., conjunctive queries, positive queries, positive relational algebra)

Definition ($\mathcal{L}$-Equivalence)

Two incomplete databases $\mathcal{I}$, $\mathcal{J}$ are $\mathcal{L}$-equivalent, denoted $\mathcal{I} \equiv_{\mathcal{L}} \mathcal{J}$, if for each $Q \in \mathcal{L}$ we have

$$\mathit{cert}_{\mathcal{I}}(Q) = \mathit{cert}_{\mathcal{J}}(Q)$$

That is, $\mathcal{L}$-equivalent incomplete dbs give rise to the same certain answers for all queries in $\mathcal{L}$.

Goal: For Q and $\mathbf{T}$, find a $\mathbf{T}'$ such that $\mathbf{T}'$ is $\mathcal{L}$-equivalent to $Q(\mathit{Rep}(\mathbf{T}))$,
for a suitable $\mathcal{L}$

Weak Representation Systems (cntd)

$\mathcal{L}^{+}_{\text{calc}}$ language of positive relational calculus queries

Theorem (Imielinski/Lipski)

For every positive query Q and v-table $\mathbf{T}$, one can compute a v-table $\mathbf{T}'$ such that

$$\mathit{Rep}(\mathbf{T}') \equiv_{\mathcal{L}^{+}_{\text{calc}}} Q(\mathit{Rep}(\mathbf{T}))$$

Proof.

Apply Q to $\mathbf{T}$, treating variables like constants. □

That is, $\mathbf{T}'$

- contains enough information to compute certain answers to positive queries on $Q(\mathit{Rep}(\mathbf{T}))$
- can be considered the answer of Q over $\mathbf{T}$, in the context of positive queries

Source Descriptions in GLAV

GLAV combines the approaches "global as view" and "local as view"

The components are two **schemas**

- $\mathcal{G}$, the *domain* or *global schema* ($R \in \Sigma_{\mathcal{G}}$, or $R \in \mathcal{G}$ *(by abuse of notation)*)
- $\mathcal{L}$, the *source* or *local schema* ($S \in \Sigma_{\mathcal{L}}$, or $S \in \mathcal{L}$ *(...)*)

and two sets of **views** (= relations defined by queries)

- $\mathcal{W}$, the *domain* or *global views* ($W \in \mathcal{W}$)
- $\mathcal{V}$, the *source* or *local views* ($V \in \mathcal{V}$)

Here: no assumptions about the query languages of the views

Later: Investigate the effects of the choice of language

Information Integration System (formally ...)

- An **information integration system** (IIS) $\mathcal{I} = (\mathcal{G}, \mathcal{L}, \mathcal{M})$ is given by two **schemas** $\mathcal{G}$ and $\mathcal{L}$ and a set $\mathcal{M}$ of **mappings**

$$V \subseteq W \qquad \text{or} \qquad V = W$$

  involving source views V and domain views W
- A *domain instance* $\mathbf{J}$ interprets symbols $R \in \mathcal{G}$ and domain views $W \in \mathcal{W}$ as relations $\mathbf{J}(R)$ and $\mathbf{J}(W)$, resp.
- A *source instance* $\mathbf{I}$ interprets symbols $S \in \mathcal{L}$ und source views $V \in \mathcal{V}$ as relations $\mathbf{I}(S)$ and $\mathbf{I}(V)$, resp.
- A *domain instance* $\mathbf{J}$ is **compatible** with a *source instance* $\mathbf{I}$ if

$$\mathbf{I}(V) \subseteq \mathbf{J}(W) \qquad \text{or} \qquad \mathbf{I}(V) = \mathbf{J}(W),$$

  for every constraint $V \subseteq W$ or $V = W$, resp.

Special Case "Global as View"

- Domain views = global relations, that is, $W_R(\bar{x})$:- $R(\bar{x})$
- per domain relation, there is *exactly one* source view, that is,

$$\mathcal{M} = \{V_R\ \rho_R\ R \mid R \in \mathcal{G}\} \qquad \text{where } \rho_R \in \{\subseteq, =\}$$

"$V_R \subseteq R$": the mapping of R is **sound**

"$V_R = R$": the mapping of R is **exact**

Notation: Given a source instance $\mathbf{I}$, we define

$$\mathcal{V}(\mathbf{I}) := \{R(\bar{t}\,) \mid t \in V_R(\mathbf{I}),\ R \in \mathcal{G}\},$$

the tuples mapped from $\mathbf{I}$ by the local views $\mathcal{V}$ to the global schema $\mathcal{G}$

Observation: $\mathbf{J}$ is compatible with $\mathbf{I}$ iff

$\mathcal{V}(\mathbf{I}) \subseteq \mathbf{J}$ if all mappings are *sound*

$\mathcal{V}(\mathbf{I}) = \mathbf{J}$ if all mappings are *exact*

Special Case "Local as View"

- Source views = local relations, that is, $V_S(\bar{x})$:– $S(\bar{x})$
- per source relation, there is *exactly one* domain view, that is,

$$\mathcal{M} = \{S\ \rho_S\ W_S \mid S \in \mathcal{L}\} \qquad \text{where } \rho_S \in \{\subseteq, =\}$$

"$S \subseteq W_S$": the mapping of R is **sound**

"$S = W_S$": the mapping of R is **exact**

Notation: Given a domain instance $\mathbf{J}$, we define

$$\mathcal{W}(\mathbf{J}) := \{S(\bar{t}\,) \mid t \in W_S(\mathbf{J}),\ S \in \mathcal{L}\},$$

the tuples mapped from $\mathbf{J}$ by the global views $\mathcal{W}$ to the local schema $\mathcal{L}$

Observation: $\mathbf{J}$ is compatible with $\mathbf{I}$ iff

$\mathbf{I} \subseteq \mathcal{W}(\mathbf{J})$ if all mappings are *sound*

$\mathbf{I} = \mathcal{W}(\mathbf{J})$ if all mappings are *exact*

Certain Answers

Let $\mathcal{M} = \{V_i\ \rho_i\ W_i \mid i = 1, \ldots, n\}$ be the set of mappings of an IIS

$\mathbf{I}$ a source instance

Q a query over $\mathcal{G}$ $\quad$ (= the global schema)

Definition (Certain Answers)

A tuple $\bar{d}$ ist a **certain answer** for Q w.r.t. $\mathbf{I}$ if

$$\bar{d} \in Q(\mathbf{J}) \quad \text{for alle } \mathbf{J} \text{ compatible with } \mathbf{I}.$$

The set of all certain answers for Q w.r.t. $\mathbf{I}$ is denoted as

$$\mathit{cert}_{\mathbf{I}}(Q)$$

Proposition

$$\mathit{cert}_{\mathbf{I}}(Q) \quad = \bigcap_{\mathbf{J} \text{ compatible with } \mathbf{I}} Q(\mathbf{J})$$

Certain Answers under GAV

Let $\mathcal{M} = \{V_R \subseteq/= R \mid R \in \mathcal{G}\}$ be a set of GAV mappings

$\mathbf{I}$ a source instance

Q a query over $\mathcal{G}$ (= the global schema)

$\leadsto$ When is a global instance $\mathbf{J}$ compatible with $\mathbf{I}$?

Exact Mappings: $\mathcal{V}(\mathbf{I}) = \mathbf{J}$

$\Rightarrow$ *only one* instance is compatible!

$\Rightarrow$ $\mathit{cert}_{\mathbf{I}}(Q) = Q(\mathcal{V}(\mathbf{I}))$

Sound Mappings: $\mathcal{V}(\mathbf{I}) \subseteq \mathbf{J}$

$\Rightarrow$ supersets of $\mathcal{V}(\mathbf{I})$ are compatible

$\leadsto$ do we still have $\mathit{cert}_{\mathbf{I}}(Q) = Q(\mathcal{V}(\mathbf{I}))$?

GAV with sound mappings: $\mathbf{J}$ compatible with $\mathbf{I}$ iff $\mathcal{V}(\mathbf{I}) \subseteq \mathbf{J}$

GAV with Exact Mappings

Definition (Monotonic Query)

A query Q is **monotonic** if for all instances $\mathbf{I}_1$, $\mathbf{I}_2$ we have

$$\mathbf{I}_1 \subseteq \mathbf{I}_2 \quad \Rightarrow \quad Q(\mathbf{I}_1) \subseteq Q(\mathbf{I}_2)$$

- Datalog (= Horn clauses w/o function symbols) queries are monontonic
- Queries with negation are in general not monotonic

Proposition

Consider an IIS with exact GAV mappings and let Q be a query. Then:

$$Q \text{ monotonic} \quad \Rightarrow \quad \mathit{cert}_{\mathbf{I}}(Q) = Q(\mathcal{V}(\mathbf{I}))$$

If $\mathit{cert}_{\mathbf{I}}(Q) = Q(\mathcal{V}(\mathbf{I}))$, then we can compute the certain answers for Q by evaluating $Q' = Q \circ \mathcal{V}$ on the source instance $\mathbf{I}$

$\leadsto$ Q' is a "query plan" for Q

How Difficult is Finding Certain Answers under Exact LAV?

Example: LAV with exact mappings (Abiteboul/Duschka)

1. Global relations model a coloured graph:

$\texttt{Edge}(X,Y)$: there is an edge from vertex X to vertex Y

$\texttt{Colour}(X,Z)$: vertex X has colour Z

2. Source relations S_1, S_2, S_3 are *mapped exactly* by domain views W_1, W_2, W_3

$$\mathcal{M} = \{S_1 = W_1,\ S_2 = W_2,\ S_3 = W_3\},$$

where

$$\begin{aligned} W_1(X) &\coloneq \texttt{Colour}(X,Y) \\ W_2(Y) &\coloneq \texttt{Colour}(X,Y) \\ W_3(X,Y) &\coloneq \texttt{Edge}(X,Y). \end{aligned}$$

Thus, we have the vertices in S_1, the colours in S_2, the edges in S_3

Certain Answers under LAV? (Cont)

3. Source Instances.

Graph $G = (V, E)$ *(V are the vertices, E the edges)*

Define the source instance $\mathbf{I}_G$ by

$$\begin{aligned}\mathbf{I}_G(S_1) &:= V\\ \mathbf{I}_G(S_2) &:= \{\texttt{red},\ \texttt{green},\ \texttt{blue}\}\\ \mathbf{I}_G(S_3) &:= E.\end{aligned}$$

4. Compatible Instances.

A global instance $\mathbf{J}$ is compatible with $\mathbf{I}_G$ if

- $\mathbf{J}(\texttt{Edge})$ contains exactly the edges in E
- $\mathbf{J}(\texttt{Colour})$ assigns to the vertices of G the colours `red`, `green`, `blue`

Certain Answers under LAV? (Cont)

5. Query.

$$Q() \;{:}\!-\; \texttt{Edge}(X,Y),\ \texttt{Colour}(X,Z),\ \texttt{Colour}(Y,Z)$$

Q returns the answer $()$ over $\mathbf{J}$ if and only if

$\mathbf{J}$ contains neighbouring vertices X, Y with the same colour

6. Certain Answers.

Observe, "$()$" is a certain answer for Q wrt $\mathbf{I}_G$ iff

every colouring of G with three colours

assigns the some colour to two neighbouring vertices

Thus: G is not 3-colourable iff $\mathit{cert}_{\mathbf{I}_G}(Q) = \{()\}$

Certain Answers under LAV? (Cont)

3-Colorability is NP-complete

7. Conclusion.
To decide whether a tuple is a certain answer under LAV is coNP-hard, if sources are mapped **exactly**.

This holds already for

- relational conjunctive queries and
- views defined by relational conjunctive queries.

And what if the sources are not mapped exactly?

Computing Certain Answers under LAV

GAV:

- certain answers for Q can in general be computed by evaluating a query Q' over the sources
- Q' results from Q by a simple transformation

$\rightsquigarrow$ is that also possible for LAV?

Problem with LAV and *exact* mappings:

If: $\mathit{cert}_{\mathbf{I}}(Q)$ can be computed by evaluating a query Q' over the sources

Then: the problem "$\bar{d} \in \mathit{cert}_{\mathbf{I}}(Q)$" is tractable (for a fixed Q)

(Evaluation of Datalog or PL1 queries is polynomial)

But: there is a conjunctive set of mappings $\mathcal{M}$ und a conjunctive query Q, such that "$\bar{d} \in \mathit{cert}_{\mathbf{I}}(Q)$" is coNP-hard

GAV and LAV

The approach for GAV was:

- find prototypical database instance $\mathbf{J}_0$
- evaluate Q over $\mathbf{J}_0 \quad \rightsquigarrow \quad \mathit{cert}_{\mathbf{I}}(Q)$

To LAV, this can only be applied if mappings are sound, but not exact:

- $\mathcal{M} = \{S_i \subseteq W_i \mid S_i \in \mathcal{S}\}$

$\rightsquigarrow$ $\mathbf{J}$ compatible with $\mathbf{I}$ iff $\mathbf{I} \subseteq \mathcal{W}(\mathbf{J})$

- Can we invert $\mathcal{W}$ to $\mathcal{W}^{-1}$?

$\rightsquigarrow$ If so, a compatible $\mathbf{J}$ would have to satisfy $\mathcal{W}^{-1}(\mathbf{I}) \subseteq \mathbf{J}$

Inverse Rules: Idea (1)

Example: global relation Edge, sources $S_1 \subseteq W_1$, $S_2 \subseteq W_2$ where

$$W_1(X) \mathrel{:\!-} \texttt{Edge}(X, Z)$$
$$W_2(X, Y) \mathrel{:\!-} \texttt{Edge}(X, Z) \wedge \texttt{Edge}(Z, Y)$$

Let $\mathbf{J}$ be defined as

$$\mathbf{J}(\texttt{Edge}) = \{\langle a, b\rangle,\ \langle b, c\rangle,\ \langle c, d\rangle,\ \langle d, e\rangle\}$$

Let $\mathbf{I} := \mathcal{W}(\mathbf{J})$, that is,

$$\mathbf{I}(S_1) = \{a,\ b,\ c,\ d\}$$
$$\mathbf{I}(S_2) = \{\langle a, c\rangle,\ \langle b, d\rangle,\ \langle c, e\rangle\}$$

How far can we reconstruct $\mathbf{J}$ from $\mathbf{I}$?

Inverse Rules: Idea (2)

In W_1, W_2, there are existential variables

$\Rightarrow$ a compatible $\mathbf{J}$ must contain elements for these

Idea: Generate lost elements by Skolem functions

$$W_1(X) \coloneq \texttt{Edge}(X, Z)$$
$$W_2(X, Y) \coloneq \texttt{Edge}(X, Z) \wedge \texttt{Edge}(Z, Y)$$

$\rightsquigarrow$ Inverse rules $\mathcal{W}^{-1}$ for `Edge`:

$$\texttt{Edge}(X, f(X)) \coloneq S_1(X)$$
$$\texttt{Edge}(X, g(X, Y)) \coloneq S_2(X, Y)$$
$$\texttt{Edge}(g(X, Y), Y) \coloneq S_2(X, Y)$$

Inverse Rules: Definition

Let the conjunctive domain view in the mapping $S \subseteq W$ be defined by

$$W(\bar{x}) \mathrel{:}\!- R_1(\bar{s}_1),\ \ldots,\ R_n(\bar{s}_n)$$

The inverse rules for W are

$$R_j(\bar{t}_j) \mathrel{:}\!- S(\bar{x}), \qquad j = 1, \ldots, n$$

where $\bar{t}_j$ originates from $\bar{x}_j$ as follows:

- constants und distinguished variables from $\bar{x}$ stay unchanged
- if $x \in \bar{s}_j$ is the i-th existential variable, say z_i,
  then x is replaced by Skolem term $f_i^S(\bar{x})$

Observation: for a collection of *conjunctive* views $\mathcal{W}$
the set of rules $\mathcal{W}^{-1}$ is *not* recursive

Inverse Rules: Example

For $\mathbf{J}_0 := \mathcal{W}^{-1}(\mathbf{I})$ we have

$$\begin{aligned}\mathbf{J}_0(\texttt{Edge}) = \{&\langle a, f(a)\rangle,\ \langle b, f(b)\rangle,\ \langle c, f(c)\rangle,\ \langle d, f(d)\rangle,\\ &\langle a, g(a,c)\rangle,\ \langle b, g(b,d)\rangle,\ \langle c, g(c,e)\rangle,\\ &\langle g(a,c), c\rangle,\ \langle g(b,d), d\rangle,\ \langle g(c,e), e\rangle\}\end{aligned}$$

Query: $Q(X,Y) \mathrel{:\!-} \texttt{Edge}(X,Z_1),\ \texttt{Edge}(Z_1,Y),\ \texttt{Edge}(Y,Z_2)$

Result: $Q(\mathbf{J}_0) = \{\langle a, c\rangle,\ \langle b, d\rangle,\ \langle g(a,c),\ g(c,e)\rangle\}$

What happens for

$$\begin{aligned} Q_1(X,Y) &\mathrel{:\!-} \texttt{Edge}(X,Z),\ \texttt{Edge}(Z,Y)\\ Q_2(X,Y) &\mathrel{:\!-} \texttt{Edge}(X,Y),\ \texttt{Edge}(Y,Z)\\ Q_3(X,Y) &\mathrel{:\!-} \texttt{Edge}(X,Y)\\ Q_3(X,Y) &\mathrel{:\!-} \texttt{Edge}(X,Z),\ Q_3(Z,Y) \quad ?\end{aligned}$$

Inverse Rules: Idea (3)

Observation: In the examples, $Q(\mathcal{W}^{-1}(\mathbf{I}))$ returned certain answers
. . . and more

Idea: compute $Q(\mathcal{W}^{-1}(\mathbf{I}))$ — and remove the tuples with Skolem terms

Definition (Cutting out Skolem Terms)

$$(Q \circ \mathcal{W}^{-1})^{\downarrow}(\mathbf{I}) = \{\bar{t} \in Q(\mathcal{W}^{-1}(\mathbf{I})) \mid \bar{t} \text{ contains no Skolem term}\}$$

$Q \circ \mathcal{W}^{-1}$ can itself be seen as a query:

$$\text{Rules for } Q \circ \mathcal{W}^{-1} \ = \ \text{Rules for } Q \ \cup \text{inverse rules}$$

Question (to be addressed later on):

Can we express $(Q \circ \mathcal{W}^{-1})^{\downarrow}$ as a conjunctive query?

Inverse Rules and Certain Answers

Proposition

$\mathcal{W}^{-1}(\mathbf{I})$ is compatible with $\mathbf{I}$

Proof.

Let $\mathbf{J}_0 := \mathcal{W}^{-1}(\mathbf{I})$. We show that $\mathbf{I} \subseteq \mathcal{W}(\mathbf{J}_0)$.
Let S be a source relation and $\bar{d} \in \mathbf{I}(S)$.
Suppose the domain view W_S in the mapping "$S \subseteq W_S$" $\in \mathcal{M}$ is defined as

$$W_S(\bar{x}) \coloneq R_1(\bar{s}_1), \ldots, R_n(\bar{s}_n).$$

The inverse rules are $R_i(\bar{t}_i) \coloneq S(\bar{x})$.

For $\bar{d}$ the inverse rules generate the tuples $\bar{t}'_i := [\bar{x}/\bar{d}]\bar{t}_i \in \mathbf{J}_0(R_i)$,
which originate from the $\bar{t}_i$, by replacing the x_j with d_j.

For the assignment $\alpha = [x_1/d_1, \ldots, x_n/d_k,\ z_1/f_1^S(\bar{d}), \ldots, z_m/f_m^S(\bar{d})]$,
we have $\mathbf{J}_0 \models \alpha(R_i(\bar{s}_i))$.

Thus, application of the rule for W_S gives $\bar{d} \in W_S(\mathbf{J}_0)$. $\square$

Inverse Rules and Certain Answers

Corollary (Completeness)

Let $\mathcal{W}$ be the set of domain views describing the sources in a set of sound LAV mappings. Then

$$\mathit{cert}_{\mathbf{I}}(Q) \subseteq (Q \circ \mathcal{W}^{-1})^{\downarrow}(\mathbf{I})$$

for **all queries** Q.

Proof.

$\mathcal{W}^{-1}(\mathbf{I})$ compatible with $\mathbf{I}$ $\Rightarrow$ $\mathit{cert}_{\mathbf{I}}(Q) \subseteq Q(\mathcal{W}^{-1}(\mathbf{I}))$

No certain answer contains Skolem terms $\Rightarrow$ $\mathit{cert}_{\mathbf{I}}(Q) \subseteq (Q \circ \mathcal{W}^{-1})^{\downarrow}(\mathbf{I})$ □

Inverse Rules and Certain Answers/2

Theorem (Soundness)

Let $\mathcal{W}$ be the set of domain views describing the sources in a set of sound LAV mappings. Then

$$(Q \circ \mathcal{W}^{-1})^{\downarrow}(\mathbf{I}) \subseteq \mathit{cert}_{\mathbf{I}}(Q)$$

for **all relational conjunctive queries** Q.

Proof will be added later. Uses the Universal Model Lemma below.

Inverse Rules and Certain Answers/3

$\mathcal{W}^{-1}$ contains in its domain elements (Skolem terms) that are not in **dom**

Let **sko** be the set of all Skolem terms.

Let $\mathbf{J}$ be a "normal" instance and $\mathbf{J}'$ an instance over **dom** $\cup$ **sko**.

A **homomorphism** from $\mathbf{J}'$ to $\mathbf{J}$ is a mapping η: **sko** $\rightarrow$ **dom** such that $\eta A \in \mathbf{J}$ for every atom $A \in \mathbf{J}'$, that is $\eta R(\bar{t}) = R(\eta\bar{t}) \in \mathbf{J}$, whenever $R(\bar{t}) \in \mathbf{J}'$.

Remark

If we view Skolem terms as variables, then $\mathbf{J}'$ is a v-(multi-)table.

In this perspective, there is a homomorphism from $\mathbf{J}'$ to $\mathbf{J}$ iff $\mathbf{J} \in \mathit{Rep}(\mathbf{J}')$.

Inverse Rules and Certain Answers/4

Lemma (Universal Model)

Let $\mathcal{I} = (\mathcal{G}, \mathcal{L}, \mathcal{M})$ be with sound LAV mappings and conjunctive views $\mathcal{W}$.
Let $\mathbf{I}$ be a source instance and $\mathbf{J}$ be a global instance.
Then the following are equivalent:

1. $\mathbf{J}$ is compatible with $\mathbf{I}$ (wrt $\mathcal{I}$)
2. there is a homomorphism from $\mathcal{W}^{-1}(\mathbf{I})$ to $\mathbf{J}$

Proof will be added later.

Query Plans: Definition

It would be nice to compute the certain answers for Q (or as many as possible) by running a (simple) query P on the sources.

Such a P could be considered a *logical plan* for answering Q

Definition

A query P over the source schema $\mathcal{L}$ is a **logical query plan** for Q if

$$P(\mathbf{I}) \subseteq \mathit{cert}_{\mathbf{I}}(Q)$$

for all source instances $\mathbf{I}$.

How can one recognize that P is a query plan for Q?

$\rightsquigarrow$ Theory of query equivalence and containment

Containment and Equivalence Modulo a set of Views

$\mathcal{G}$ global schema, $\mathcal{W}$ set of views over $\mathcal{G}$

P query over $\mathcal{L}$, Q query over $\mathcal{G}$

Definition

P is **contained in** Q **modulo** $\mathcal{W}$, denoted $P \subseteq_{\mathcal{W}} Q$, iff

$$P(\mathcal{W}(\mathbf{J})) \subseteq Q(\mathbf{J})$$

for all instances $\mathbf{J}$ of $\mathcal{G}$

This means:

- We extend all $\mathbf{J}$, using $\mathcal{W}$, so that the source relations $S \in \mathcal{L}$ are interpreted, too. Call the extensions $\mathbf{J}_{\mathcal{W}}$
- Then check "$P(\mathbf{J}_{\mathcal{W}}) \subseteq Q(\mathbf{J}_{\mathcal{W}})$" for all $\mathbf{J}$

Analogously: P is equivalent to Q modulo $\mathcal{W}$, denoted $P \equiv_{\mathcal{W}} Q$

Query Plans and Containment Modulo a set of Views

Proposition (Plans are Contained)

If P is a plan for Q, then $P \subseteq_{\mathcal{W}} Q$.

Proof.

If $\mathbf{J}$ is a global instance, then $\mathcal{W}(\mathbf{J})$ is a source instance and $\mathbf{J}$ is compatible with $\mathcal{W}(\mathbf{J})$.

Thus: $P(\mathcal{W}(\mathbf{J})) \subseteq \mathit{cert}_{\mathcal{W}(\mathbf{J})}(Q) \subseteq Q(\mathbf{J})$. □

Query Plans and Containment Modulo a Set of Views

Proposition (Monotonic Containees are Plans)

Let P be monotonic. Then

$$P \subseteq_{\mathcal{W}} Q \quad \Rightarrow \quad P \text{ is a plan for } Q$$

Proof.

Let $\mathbf{I}$ be a source instance. We show that $P(\mathbf{I}) \subseteq \mathit{cert}_{\mathbf{I}}(Q)$.

Let $\mathbf{J}$ be compatible with $\mathbf{I}$ $\Rightarrow$ $\mathcal{W}(\mathbf{J})$ is a source instance with $\mathbf{I} \subseteq \mathcal{W}(\mathbf{J})$.

P monotonic $\Rightarrow$ $P(\mathbf{I}) \subseteq P(\mathcal{W}(\mathbf{J}))$.

$P \subseteq_{\mathcal{W}} Q$ $\Rightarrow$ $P(\mathcal{W}(\mathbf{J})) \subseteq Q(\mathbf{J})$. Hence: $P(\mathbf{I}) \subseteq Q(\mathbf{J})$

$\mathbf{J}$ was arbitrary $\Rightarrow$ $P(\mathbf{I}) \subseteq \bigcap_{\mathbf{J} \text{ compatible with } \mathbf{I}} Q(\mathbf{J}) = \mathit{cert}_{\mathbf{I}}(Q)$

□

Query Plans and Containment Modulo a Set of Views/2

Proposition (Exact Mappings)

Suppose all LAV mappings in $\mathcal{W}$ are exact, Q is a query over the global schema, and P is a query over the sources. Then

$$P \subseteq_{\mathcal{W}} Q \quad \Rightarrow \quad P \text{ is a plan for } Q$$

Proof.

Let $\mathbf{I}$ be a source instance. We show that $P(\mathbf{I}) \subseteq \mathit{cert}_{\mathbf{I}}(Q)$
$\mathbf{J}$ is a global instance compatible with $\mathbf{I} \;\Rightarrow\; \mathcal{W}(\mathbf{J}) = \mathbf{I}$
$P \subseteq_{\mathcal{W}} Q \;\Rightarrow\; P(\mathbf{I}) = P(\mathcal{W}(\mathbf{J})) \subseteq Q(\mathbf{J})$.
As before, this shows $P(\mathbf{J}) \subseteq \mathit{cert}_{\mathbf{I}}(Q)$ □

Thus: in the case of *monotonic plans* or *exact mappings*,
logical query plans are characterized by "containment module $\mathcal{W}$"
$\leadsto$ how can we recognize "containment modulo $\mathcal{W}$"?
$\leadsto$ how can we generate plans for Q?

Reduction "$\subseteq_{\mathcal{W}}$" $\rightarrow$ "$\subseteq$"

Let P be a plan for Q

If the views in $\mathcal{W}$ are not recursive,
we can **unfold** the relation symbols of the views occurring in P ,
that is, we can replace them by their definitions

Notation: P^{unf} is the *unfolding* of P

Clearly: $P \equiv_{\mathcal{W}} P^{unf}$

Consequence: $P \subseteq_{\mathcal{W}} Q$ iff $P^{unf} \subseteq Q$

What can we say about the difficulty of checking whether P is a plan of Q?

Unfolding Example (A. Halevy)

Global Relations

$\texttt{Cites}(x, y)$ if x cites y

$\texttt{SameTopic}(x, y)$ if x and y work on the same topic

Query

$Q(x, y) \coloneq \texttt{SameTopic}(x, y), \texttt{Cites}(x, y), \texttt{Cites}(y, x)$

Global Views, describing two sources

$W_1(u, v) \coloneq \texttt{Cites}(u, v), \texttt{Cites}(v, u)$

$W_2(u, v) \coloneq \texttt{SameTopic}(u, v), \texttt{Cites}(u, u'), \texttt{Cites}(v, v')$

Suggested Plan

$P(x, y) \coloneq W_1(x, y), W_2(x, y)$

More Questions About Plans

- Can all certain answers be computed by plans?
- How many plans do we need?
- How can we compare plans?
- Is there a best set of plans?
- If so, how can we find it?

In LAV, the Certain Answer Function is Monotonic

We note that for sound LAV mappings, the function

$$\mathbf{I} \mapsto \mathit{cert}_{\mathbf{I}}(Q)$$

is always monotonic

Proposition

Consider an IIS with sound LAV mappings and let Q be any query. Then

$$\mathbf{I} \subseteq \mathbf{I}' \qquad \Rightarrow \qquad \mathit{cert}_{\mathbf{I}}(Q) \subseteq \mathit{cert}_{\mathbf{I}'}(Q)$$

The same holds for GLAV systems where the source views are monotonic

Logical Plans and Certain Answers

Proposition

Let $\mathcal{W}$ and Q be arbitrary.
For every $\mathbf{I}$ and $d \in \mathit{cert}_{\mathbf{I}}(Q)$ there exists a conjunctive plan P for Q such that

$$\bar{d} \in P(\mathbf{I})$$

Proof.

Suppose $\mathbf{I}(S_i) = \{\bar{d}_{i,1},\ \ldots, \bar{d}_{i,n_i}\}$ for $i \in [1,k]$

As on an earlier occasion, define P as

$$P(\bar{d}) \coloneq W_1(\bar{d}_{1,1}), \ldots, W_1(\bar{d}_{1,n_1}), \ldots, W_k(\bar{d}_{k,1}), \ldots, W_k(\bar{d}_{k,n_k})$$

Since $\bar{d} \in P(\mathbf{I})$, we only need to show that P is a plan for Q, that is, $P \subseteq_{\mathcal{W}} Q$.

Let $\mathbf{J}$ be a global instance.

Case 1: $\mathbf{I} \subseteq \mathcal{W}(\mathbf{J}) \ \Rightarrow\ P(\mathcal{W}(\mathbf{J})) = \{\bar{d}\} \subseteq Q(\mathbf{J})$, since $\bar{d}$ is a certain answer

Case 2: $\mathbf{I} \not\subseteq \mathcal{W}(\mathbf{J}) \ \Rightarrow\ P(\mathcal{W}(\mathbf{J})) = \emptyset \subseteq Q(\mathbf{J})$ $\square$

Complete Sets of Plans

Let $\mathcal{W}$ be a set of global views and Q be a query.

Then $\boxed{\textsc{Plans}_{\mathcal{W}}(Q)}$ denotes the **set of all conjunctive query plans** for Q in the IIS with sound mappings defined by $\mathcal{W}$.

Definition

- A subset $\mathcal{P} \subseteq \textsc{Plans}_{\mathcal{W}}(Q)$ is **complete** if for every source instance $\mathbf{I}$ and every certain answer $\bar{d} \in \mathit{cert}_{\mathbf{I}}(Q)$, there is a $P \in \mathcal{P}$ such that $\bar{d} \in P(\mathbf{I})$
- A complete set $\mathcal{P}$ is **minimal** if no proper subset is complete.

Let $\mathcal{P}$ be a complete set of plans. Then for every Q and $\mathbf{I}$ we have

$$\mathit{cert}_{\mathbf{I}}(Q) = \bigcup_{P \in \mathcal{P}} P(\mathbf{I})$$

Do miminal complete sets of plans exist? What is their size?

Covering Sets of Plans

Definition

- $\mathcal{P} \subseteq \textsc{Plans}_{\mathcal{W}}(Q)$ is **covering** if for every plan P' there are plans $P_1, \ldots, P_n \in \mathcal{P}$ such that $P' \subseteq P_1 \cup \cdots \cup P_n$
- $\mathcal{P} \subseteq \textsc{Plans}_{\mathcal{W}}(Q)$ is **dominating** if for every plan P' there is a plan $P \in \mathcal{P}$ such that $P' \subseteq P$
- A covering (dominating) set is **minimal** if no proper subset is covering (dominating)
- Plan P is **maximal** if for every plan P' we have $\quad P \subseteq P' \;\Rightarrow\; P' \subseteq P$

Proposition

Let $\mathcal{P}$ be dominating set of plans and P be a maximal plan.
Then $\mathcal{P}$ contains a plan P' such that $P \equiv P'$.

How are complete, covering, and dominating sets of plans related?

Plans in the Relational Case

On this and the next slide, we assume that $\mathcal{W}$ and Q are relational and we consider only relational plans.

Theorem (Covering by Maximal Plans)

1. A covering set of plans is dominating.
2. A minimal covering set contains only maximal plans.

Proof.

Claim 1 holds because for relational conjunctive queries we have that $Q \subseteq Q_1 \cup \cdots \cup Q_n$ iff $Q \subseteq Q_i$ for some $i \in [1, n]$.

Claim 2 holds for all dominating sets in preorders. $\square$

Note that Claim 1 would not hold for conjunctive queries with disequations or comparisons

Plans in the Relational Case/2

Theorem (Maximal Plans are Small and Simple)

Let $P \in \textsc{Plans}_{\mathcal{W}}(Q)$ be maximal. Then

1. P has at most as many atoms as Q
2. P contains only constants occurring in Q or in $\mathcal{W}$

Proof.

Both claims follow from that fact that P is a plan iff $P^{unf} \subseteq Q$ iff there is a homomorphism from Q to P^{unf}. □

The last two theorems tells us how we can compute, in principle, a minimal dominating (= covering) set of plans.

I am note aware that anyone has shown how difficult it is do decide whether a query over the sources is a maximal plan.

Questions about Logical Plans

- Given a set of views $\mathcal{W}$, how many maximal plans for Q are there? At most? At least?
- Is it also possible in an exact LAV setting to compute all certain answers by plans?
- What is the data complexity of deciding certain answers
  - in a sound LAV setting?
  - in an exact LAV setting?
- What can we say about the difficulty of computing certain answers in a sound LAV setting if
  - the query can contain comparisons?
  - the views can contain comparisons?

Plans and Rewriting Queries Using Views

The problem of computing logical query plans in a sound LAV setting is the same as the one to compute *rewritings* of a query Q *using views* $\mathcal{W} = \{W_1, \ldots, W_n\}$.

A query R over the relations in $\mathcal{W}$ is a (contained) **rewriting** of Q if

$$R^{unf} \subseteq Q.$$

It is an **exact** rewriting if

$$R^{unf} \equiv Q.$$

All results about covering, domininating, maximal plans etc. can be rephrased as results about rewritings.

The "Bucket" Algorithm

The Bucket Algorithm was developed to generate query plans for the *Information Manifold* system, the first LAV integration system [Levy/Rajaraman/Ordille 1996].

Goal: Given a conjunctive query Q, compute a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of plans for Q

If Q is relational, we want $\mathcal{P}$ to be covering wrt. "$\subseteq$"

(i.e., for every plan P for Q there is a P_i with $P \subseteq P_i$)

The "Bucket"-Algorithm in an Example

Global schema:

$\texttt{Registered}(\texttt{student}, \texttt{course}, \texttt{year})$

$\texttt{Course}(\texttt{course}, \texttt{number})$

$\texttt{Enrolled}(\texttt{student}, \texttt{department})$

Sources S_1, S_2, S_3, S_4 described by the views:

$$
\begin{aligned}
W_1(s,n,y) &\mathrel{:\!-} \texttt{Registered}(s,c,y),\ \texttt{Course}(c,n),\ n \geq 500,\ y \geq 2007 \\
W_2(s,d,c) &\mathrel{:\!-} \texttt{Enrolled}(s,d),\ \texttt{Registered}(s,c,y) \\
W_3(s,c,y) &\mathrel{:\!-} \texttt{Registered}(s,c,y),\ y \leq 2005 \\
W_4(s,c,n) &\mathrel{:\!-} \texttt{Enrolled}(s,\mathsf{cs}),\ \texttt{Registered}(s,c,y), \\
&\qquad \texttt{Course}(c,n),\ n \leq 100
\end{aligned}
$$

Query: $q(S) \mathrel{:\!-} \texttt{Enrolled}(s,\mathsf{cs}),\ \texttt{Registered}(s,c,2010),$
$\texttt{Course}(c,n),\ n \geq 300$

The "Bucket"-Algorithm: 1st Step

Idea:

- for each atom in Q, collect the views that possibly can appear in a plan
- exploit: unfolded plans are homomorphic images of the query

For each relational atom $r(\bar{y})$ in the query, create a "bucket":

For atom $\boxed{r(\bar{y})}$ collect all instantiated views $W_i(\phi_i\bar{x}_i)$ such that

- $\phi_i r(\bar{z})$ occurs in the body of $W_i(\phi_i\bar{x})$
- there is a substitution θ with $\theta r(\bar{y}) = \phi_i r(\bar{z})$
  i.e., $r(\bar{y})$ and $r(\bar{z})$ are unifiable, without instantiating existential variables in W_i
- ϕ_i and θ are as general as possible
- the comparisons on the variables of the two atoms are consistent

The "Bucket"-Algorithm: the Buckets

In our example: 3 buckets

$\texttt{Enrolled}(s, \texttt{cs})$	$\texttt{Registered}(s, c, 2010)$	$\texttt{Course}(c, n)$
$W_2(s, \texttt{cs}, C')$	$W_1(s, n', 2010)$	$W_1(s', n, y')$
$W_4(s, c', n')$		

The following views do not fit into the buckets:

$W_2, W_4 \notin \text{BUCKET}(\texttt{Registered}(s, c, 2010))$: Y cannot be instantiated

$W_3 \notin \text{BUCKET}(\texttt{Registered}(s, c, 2010))$: comparisons for n are inconsistent

$W_4 \notin \text{BUCKET}(\texttt{Course}(c, n))$: comparisons for n are inconsistent

The "Bucket"-Algorithm: 2^{nd} Step

Combine the views in the buckets, 1^{st} possibility:

$$P_1(S) \ :\!-\ W_2(s, \mathtt{cs}, c'),\, W_1(s, n', 2010),\, W_1(s', n, y')$$

Unfold: $P_1^{unf}(S) :\!- \boxed{\mathtt{Enrolled}(s, \mathtt{cs})}, \mathtt{Registered}(s, c, y_1),$
$\boxed{\mathtt{Registered}(s, c_2, 2010)}, \boxed{\mathtt{Course}(c_2, n')},$
$n' \geq 500,\ 2010 \geq 2007,$
$\mathtt{Registered}(s', c_3, y'),\ \mathtt{Course}(c_3, n),$
$n \geq 500,\ y' \geq 2007$

Query: $Q(S) :\!- \boxed{\mathtt{Enrolled}(s, \mathtt{cs})}, \boxed{\mathtt{Registered}(s, c, 2010)},$
$\boxed{\mathtt{Course}(c, n)}, n \geq 300$

Clearly: there is a hom from Q to P_1^{unf} $\Rightarrow$ P_1 is a plan for Q

Moreover: P_1 is equivalent to P_1':

$$P_1'(S) :\!- W_2(s, \mathtt{cs}, c'),\, W_1(s, n', 2010)$$

The "Bucket"-Algorithm: 2^{nd} Step (cont)

Combine the views in the buckets, 2^{nd} possibility:

$$P_2(S) \ :\!-\ W_4(s, c', n'),\, W_1(s, n'', 2010),\, W_1(s', n, y')$$

Unfold: $P_2^{unf}(S)$:– $\boxed{\texttt{Enrolled}(s, \texttt{cs})}$, $\texttt{Registered}(s, c', y_1)$, $\texttt{Course}(c', n')$, $n' \leq 100$ $\boxed{\texttt{Registered}(s, c_2, 2010)}$, $\boxed{\texttt{Course}(c_2, n'')}$, $n'' \geq 500$, $2010 \geq 2007$, $\texttt{Registered}(s', c_3, y')$, $\texttt{Course}(c_3, n)$, $n \geq 500$, $y' \geq 2007$

Query: $Q(S)$:– $\boxed{\texttt{Enrolled}(s, \texttt{cs})}$, $\boxed{\texttt{Registered}(s, c, 2010)}$, $\boxed{\texttt{Course}(c, n)}$, $n \geq 300$

Clearly: there is a hom from Q to P_2^{unf} $\Rightarrow$ P_2 is a plan for Q

P_2 can be optimized analogously to P_1

Observation

The Bucket Algorithm may find exponentially many plans

Example

$Q(x_1, \ldots, x_n) \ \text{:-}\ \ r_1(x_1),\ \ldots,\ r_n(x_n)$

With $2n$ Sources S_i, S'_i, $i = 1, \ldots, n$, where

$$W_i(x_i) \ \text{:-}\ \ r_i(x_i) \qquad \text{and} \qquad W'_i(x_i) \ \text{:-}\ \ r_i(\bar{x}_i),$$

it finds 2^n plans

$$P(x_1, \ldots, x_n) \ \text{:-}\ \ \tilde{W}_1(x_1),\ \ldots,\ \tilde{W}_n(x_n), \qquad \text{where } \tilde{W}_i = W_i \text{ or } \tilde{W}_i = W'_i.$$

Note: for each plan P we have $\quad P^{\mathit{unf}} = Q$

$\Rightarrow$ all plans are equivalent wrt. "$\equiv_{\mathcal{W}}$".

However: if we drop a plan, we lose certain answers

$\leadsto$ what is the meaning of "$\equiv$"?

$\leadsto$ what does the Bucket Algorithm compute?

What does the Bucket Algorithm Compute?

Clearly: Plans for Q *(due to test $P^{unf} \subseteq Q$)*

However: The original paper [Levy/Rajaraman/Ordille 1996] does not make statements about the semantics (in particular, not about completeness)

Theorem (Grahne/Mendelzon 1999)

For relational $\mathcal{W}$ and Q, the Bucket Algorithm returns a set of plans for Q that compute all certain answers.

Even: Completeness holds as well if Q is *relational* and the views in $\mathcal{W}$ contain *comparisons* over a dense order.

Open: What does the Bucket Algorithm compute if Q contains comparisons? Under which conditions on Q is the set of plans complete?

Query Plans From Inverse Rules

Comparisons are conditions on the applicability of rules

(example only for W_1 and W_2)

$$\begin{aligned}
\texttt{Registered}(s, f_c(s,n,y), y) &:- W_1(s,n,y) \mid\mid y \geq 2007 \\
\texttt{Course}(f_c(s,n,y), n) &:- W_1(s,n,y) \mid\mid n \geq 500 \\
\texttt{Enrolled}(s,d) &:- W_2(s,d,c) \\
\texttt{Registered}(s, c, f_y(s,d,c)) &:- W_2(s,d,c)
\end{aligned}$$

Abduce the query plan from the query

$$\begin{aligned}
Q(s) :- {} & \texttt{Enrolled}(s, \texttt{cs}),\ \texttt{Registered}(s, c, 2010), \\
& \texttt{Course}(c, n),\ n \geq 300
\end{aligned}$$

$$\begin{aligned}
Q(s) :- {} & W_2(s, \texttt{cs}, c'),\ W_1(s, n', 2010), \\
& \texttt{Course}(f_c(s, n', 2010), n),\ n \geq 300
\end{aligned}$$

$$Q(s) :- W_2(s, \texttt{cs}, c'),\ W_1(s, n, 2010),\ W_1(s, n, 2010)$$

Relational Query Languages: Overview

We consider the following classes of queries:

- CQ: **relational conjunctive queries** without built-ins
- $CQ^{\leq}$: conjunctive queries **with comparisons**
- $CQ^{\neq}$: conjunctive queries **with disequations**
- UCQ: **unions** of conjunctive queries, that is, disjunctions of conjunctive queries, or non-recursive Datalog queries
- $datalog$: **Datalog queries**, that is, queries defined by (possibly recursive) rules
- FO: queries in **first-order logic,** that is, relational calculus queries

Certain Answers and Containment

Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be query languages

Let $\text{CERT}^{\textit{snd}}(\mathcal{Q}_1, \mathcal{Q}_2)$ be the **certain answer problem** for *sound source descriptions* $\mathcal{W} \subseteq \mathcal{Q}_1$ und queries $Q \in \mathcal{Q}_2$:

Given: $\mathcal{W} \subseteq \mathcal{Q}_1$, $Q \in \mathcal{Q}_2$, source instance $\mathbf{I}$ and tuple $\bar{d}$

Question: $\bar{d} \in \textit{cert}_{\mathbf{I}}(Q)$ w.r.t. $\mathcal{W}$?

Let $\text{CONT}(\mathcal{Q}_1, \mathcal{Q}_2)$ be the **containment problem** for queries in $\mathcal{Q}_1$ and $\mathcal{Q}_2$:

Given: $Q_1 \in \mathcal{Q}_1$, $Q_2 \in \mathcal{Q}_2$

Question: $Q_1 \subseteq Q_2$?

Certain Answers and Containment (cntd)

Theorem (Abiteboul/Duschka 98)

Let $\mathcal{Q}_1$, $\mathcal{Q}_2 \in \{ \ \mathit{CQ}, \ \ \mathit{CQ}^{\neq}, \ \ \mathit{PQ}, \ \ \mathit{datalog}, \ \ \mathit{FO} \ \}$. Then

- $\textsc{Cert}^{\mathit{snd}}(\mathcal{Q}_1, \mathcal{Q}_2)$ and
- $\textsc{Cont}(\mathcal{Q}_1, \mathcal{Q}_2)$

can be reduced to each other in polynomial time.

Complexity of the Containment Problem

$$\text{“}Q \subseteq Q'\text{”}$$

	Q'				
Q	*CQ*	*CQ*$^{\leq}$	*UCQ*	*datalog*	*FO*
CQ	NP	Π_2^{P}	NP	dec.	undec.
CQ$^{\leq}$	NP	Π_2^{P}	NP	dec.	undec.
UCQ	NP	Π_2^{P}	NP	dec.	undec.
datalog	dec.	undec.	dec.	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

. . . and the certain answer problem

Reduction $\textsc{cert}^{snd}(\mathcal{L}_1, \mathcal{L}_2) \to \textsc{cont}(\mathcal{L}_1, \mathcal{L}_2)$

Given Q, $\mathcal{W}$, $\mathbf{I}$ und $\bar{d}$ with $\mathbf{I}(S_i) = \{\bar{d}_{i,1}, \ \ldots, \bar{d}_{i,n_i}\}$ for $i \in [1, k]$

Define Q'' as

$$Q''(\bar{d}) \coloneq W_1(\bar{d}_{1,1}), \ldots, W_1(\bar{d}_{1,n_1}), \ldots, W_k(\bar{d}_{k,1}), \ldots, W_k(\bar{d}_{k,n_k})$$

Let $Q' := Q'' \cup \mathcal{W}$. (If $\mathcal{Q}_1$ is *CQ*, *CQ*$^{\neq}$ or *UCQ*,
then replace the view relations by their definitions.)

Show: $\bar{d} \in cert_{\mathbf{I}}(Q)$ wrt. $\mathcal{W}$ iff $Q' \subseteq Q$

"$\Rightarrow$": Let $\mathbf{J}$ be a global instance.

Case 1: $\mathbf{I} \not\subseteq \mathcal{W}(\mathbf{J}) \;\Rightarrow\; Q'(\mathbf{J}) = Q''(\mathcal{W}(\mathbf{J})) = \emptyset$

Case 2: $\mathbf{I} \subseteq \mathcal{W}(\mathbf{J}) \;\Rightarrow\; Q'(\mathbf{J}) = \{\bar{d}\} \subseteq Q(\mathbf{J})$, since $\bar{d}$ is a certain answer

Hence: $Q' \subseteq Q$

"$\Leftarrow$": Let $\mathbf{J}$ be an instance with $\mathbf{I} \subseteq \mathcal{W}(\mathbf{J}) \Rightarrow \bar{d} \in Q''(\mathbf{I}) \subseteq Q''(\mathcal{W}(\mathbf{I})) = Q'(\mathbf{I})$

$Q' \subseteq Q \;\Rightarrow\; \bar{d} \in Q(\mathbf{J})$. Hence: $\bar{d} \in cert_{\mathbf{I}}(Q)$

Reduction $\textsc{Cont}(\mathcal{Q}_1, \mathcal{Q}_2) \to \textsc{Cert}^{snd}(\mathcal{Q}_1, \mathcal{Q}_2)$

Let $Q_1 \in \mathcal{Q}_1$, $Q_2 \in \mathcal{Q}_2$

Let $\mathcal{W} := \{W\}$ be defined by Q_1 and

$$W(c) \text{ :- } Q_1(x),\ P(x), \qquad P \text{ new}$$

Define Q by Q_2 and

$$Q(c) \text{ :- } Q_2(x),\ P(x)$$

After the unfolding: $W \in \mathcal{Q}_1$, $Q \in \mathcal{Q}_2$.

Let $\mathbf{I}$ be an instance such that $\mathbf{I}(W) := \{c\}$.

Show: $Q_1 \subseteq Q_2$ iff $c \in \mathit{cert}_{\mathbf{I}}(Q)$

"$\Rightarrow$": Let $\mathbf{J}$ be a global instance with $c \in \mathcal{W}(\mathbf{J}) \ \Rightarrow\ c \in Q(\mathbf{J})$

$\Rightarrow\ c \in \mathit{cert}_{\mathbf{I}}(Q)$

"$\Leftarrow$": $Q_1 \not\subseteq Q_2 \ \Rightarrow$ for a global $\mathbf{J}$ there is some d with $d \in Q_1(\mathbf{J}) \setminus Q_2(\mathbf{J})$

W.l.o.g., $\mathbf{J}(P) = \{d\} \ \Rightarrow\ \mathbf{I} \subseteq \mathcal{W}(\mathbf{J})$ with $Q(\mathbf{J}) = \emptyset$. Thus, $c \notin \mathit{cert}_{\mathbf{I}}(Q)$