Query Processing in Data Integration Systems

Diego Calvanese

Free University of Bozen-Bolzano

BIT PhD Summer School – Bressanone
July 3–7, 2006

Structure of the course

1. Introduction to data integration
   - Basic issues in data integration
   - Logical formalization
2. Query answering in the absence of constraints
   - Global-as-view (GAV) setting
   - Local-as-view (LAV) and GLAV setting
3. Query answering in the presence of constraints
   - The role of integrity constraints
   - Global-as-view (GAV) setting
   - Local-as-view (LAV) and GLAV setting
4. Concluding remarks

Part I

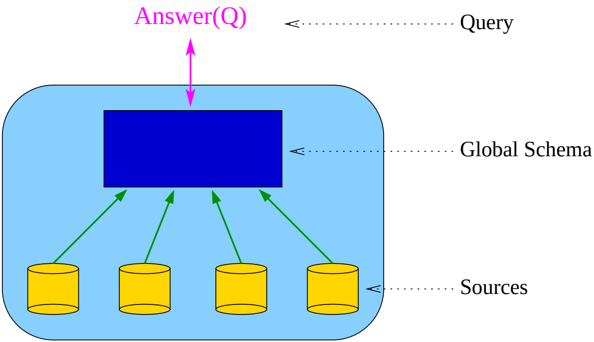
Introduction to data integration

Outline

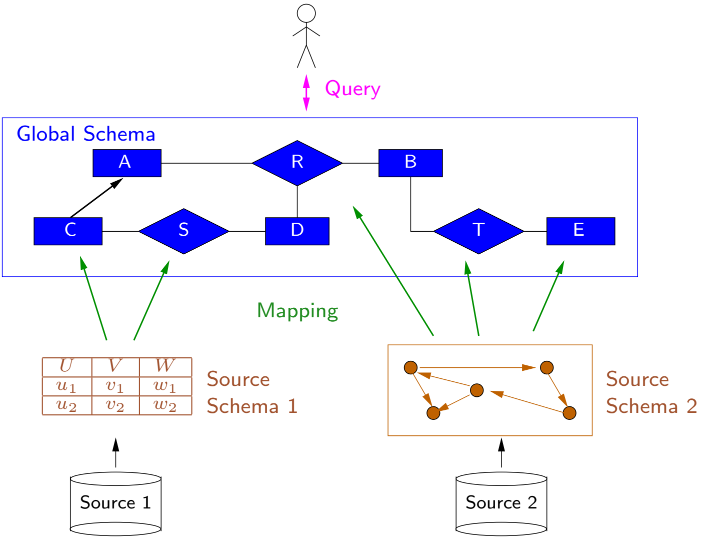
Outline

What is data integration?

Data integration is the problem of providing unified and transparent access to a collection of data stored in multiple, autonomous, and heterogeneous data sources



Conceptual architecture of a data integration system



Relevance of data integration

- Growing market
- One of the major challenges for the future of IT
- At least two contexts
  - Intra-organization data integration (e.g., EIS)
  - Inter-organization data integration (e.g., integration on the Web)

Data integration: Available industrial efforts

- Distributed database systems
- Information on demand
- Tools for source wrapping
- Tools based on database federation, e.g., DB2 Information Integrator
- Distributed query optimization

Architectures for integrated access to distributed data

- Distributed databases
  data sources are homogeneous databases under the control of the distributed database management system
- Multidatabase or federated databases
  data sources are autonomous, heterogeneous databases; procedural specification
- (Mediator-based) data integration
  access through a global schema mapped to autonomous and heterogeneous data sources; declarative specification
- Peer-to-peer data integration
  network of autonomous systems mapped one to each other, without a global schema; declarative specification

Database federation tools: Characteristics

- Physical transparency, i.e., masking from the user the physical characteristics of the sources
- Heterogeinity, i.e., federating highly diverse types of sources
- Extensibility
- Autonomy of data sources
- Performance, through distributed query optimization

However, current tools do not (directly) support logical (or conceptual) transparency

Logical transparency

Basic ingredients for achieving logical transparency:

- The global schema (ontology) provides a conceptual view that is independent from the sources
- The global schema is described with a semantically rich formalism
- The mappings are the crucial tools for realizing the independence of the global schema from the sources
- Obviously, the formalism for specifying the mapping is also a crucial point

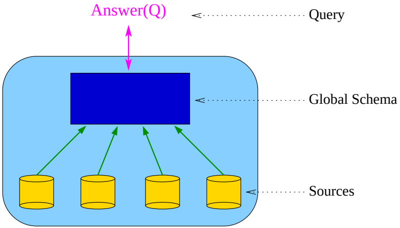
All the above aspects are not appropriately dealt with by current tools. This means that data integration cannot be simply addressed on a tool basis

Approaches to data integration

- (Mediator-based) data integration . . . *is the topic of this course*
- Data exchange [Fagin & al. TCS'05, Kolaitis PODS'05]
  - materialization of the global view
  - allows for query answering without accessing the sources
- P2P data integration [Halevy & al. ICDE'03, — & al. PODS'04, — & al. DBPL'05]
  - several peers
  - each peer with local and external sources
  - queries over one peer

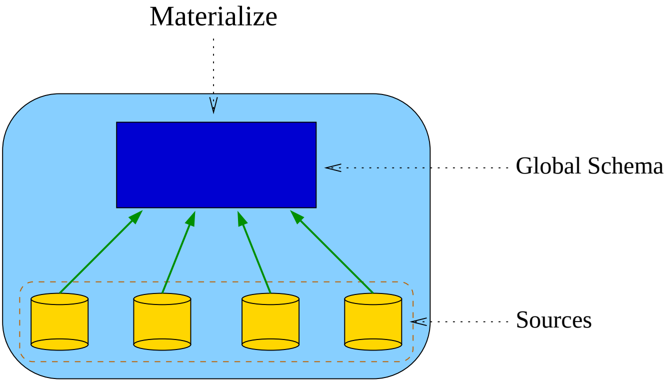
Mediator based data integration

- Queries are expressed over a global schema (a.k.a. mediated schema, enterprise model, ...)
- Data are stored in a set of sources
- Wrappers access the sources (provide a view in a uniform data model of the data stored in the sources)
- Mediators combine answers coming from wrappers and/or other mediators

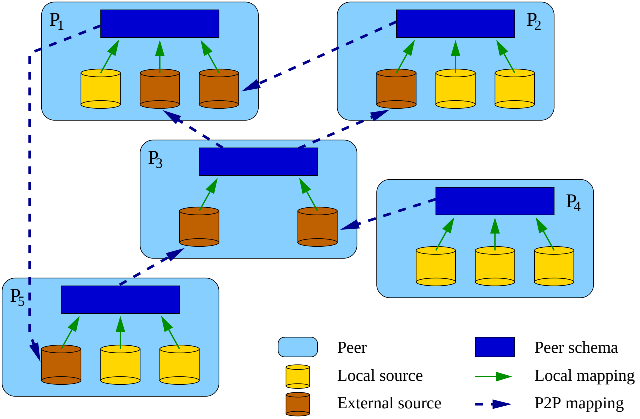


Data exchange

- Materialization of the global schema



Peer-to-peer data integration



Operations: – Answer(Q,P_i) – Materialize(P_i)

Main problems in data integration

1. How to construct the global schema
2. (Automatic) source wrapping
3. How to discover mappings between sources and global schema
4. Limitations in mechanisms for accessing sources
5. Data extraction, cleaning, and reconciliation
6. How to process updates expressed on the global schema and/or the sources ("read/write" vs. "read-only" data integration)
7. How to model the global schema, the sources, and the mappings between the two
8. How to answer queries expressed on the global schema
9. How to optimize query answering

The modeling problem

Basic questions:

- How to model the global schema
  - data model
  - constraints
- How to model the sources
  - data model (conceptual and logical level)
  - access limitations
  - data values (common vs. different domains)
- How to model the mapping between global schemas and sources
- How to verify the quality of the modeling process

A word of caution: Data modeling (in data integration) is an art.
Theoretical frameworks can help humans, not replace them

The querying problem

- A query expressed in terms of the global schema must be reformulated in terms of (a set of) queries over the sources and/or materialized views
- The computed sub-queries are shipped to the sources, and the results are collected and assembled into the final answer
- The computed query plan should guarantee
  - completeness of the obtained answers wrt the semantics
  - efficiency of the whole query answering process
  - efficiency in accessing sources
- This process heavily depends on the approach adopted for modeling the data integration system

This is the problem that we want to address in this course

Outline

Formal framework for data integration

Definition

A data integration system $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{G}$ is the global schema
  i.e., a logical theory over a relational alphabet $\mathcal{A}_{\mathcal{G}}$
- $\mathcal{S}$ is the source schema
  i.e., simply a relational alphabet $\mathcal{A}_{\mathcal{S}}$ *disjoint from* $\mathcal{A}_{\mathcal{G}}$
- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{G}$
  We consider different approaches to the specification of mappings

Semantics of a data integration system

Which are the dbs that satisfy $\mathcal{I}$, i.e., the logical models of $\mathcal{I}$?

- We refer only to dbs over a fixed infinite domain Δ of elements
- We start from the data present in the sources: these are modeled through a source database $\mathcal{C}$ over Δ (also called source model), fixing the extension of the predicates of $\mathcal{A}_{\mathcal{S}}$
- The dbs for $\mathcal{I}$ are logical interpretations for $\mathcal{A}_{\mathcal{G}}$, called global dbs

Definition

The set of databases for $\mathcal{A}_{\mathcal{G}}$ that satisfy $\mathcal{I}$ relative to $\mathcal{C}$ is:

$sem^{\mathcal{C}}(\mathcal{I}) = \{ \mathcal{B} \mid \mathcal{B}$ is a global database that is legal wrt $\mathcal{G}$ and that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ $\}$

What it means to satisfy $\mathcal{M}$ wrt $\mathcal{C}$ depends on the nature of $\mathcal{M}$

Relational calculus: the basics

Basic idea: we use the language of first-order logic to express which tuples should be in the result to a query

- We assume to have a domain Δ and a set Σ of constants, one for each element of Δ
- Let $\mathcal{A}$ be a relational alphabet, i.e., a set of predicates, each with an associated arity (we assume a positional notation)
- A database $\mathcal{D}$ over $\mathcal{A}$ and Δ is a set of relations, one for each predicate in $\mathcal{A}$, over the constants in Σ (in turn interpreted as elements of Δ)
- Let $\mathcal{L}_{\mathcal{A}}$ be the first-order language over
  - the constants in Σ
  - the predicates of $\mathcal{A}$ plus the built-in predicates of relational algebra (e.g., $<$, $>$, ...)
  - no function symbols

Relational calculus: Syntax

Definition

An (domain) relational calculus query over alphabet $\mathcal{A}$ has the form

$$\{ \ (x_1, \ldots, x_n) \ \mid \ \varphi \ \},$$

where

- $n \geq 0$ is the arity of the query
- $x_1, \ldots, x_n$ are (not necessarily distinct) variables
- φ is the body of the query, i.e., a formula of $\mathcal{L}_{\mathcal{A}}$ whose free variables are exactly $x_1, \ldots, x_n$
- $(x_1, \ldots, x_n)$ is called the target list of the query

If r is a predicate of arity k, an atom with predicate r has the form $r(y_1, \ldots, y_k)$, where $y_1, \ldots, y_k$ are variables or constants

Relational calculus: Semantics

Relational calculus queries are evaluated on particular interpretations

Definition

A correct interpretation for relational calculus queries over $\mathcal{A}$ is a pair $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$, where Δ is a domain, and $\mathcal{D}$ is a database over $\mathcal{A}$ and Δ

Definition

The value of a relational calculus query $q = \{(x_1, \ldots, x_n) \mid \varphi\}$ in an interpretation $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$ is the set of tuples $(c_1, \ldots, c_n)$ of constants in Σ such that $\langle \mathcal{I}, \mathcal{V} \rangle \models \varphi$, where $\mathcal{V}$ is the variable assignment that assigns c_i to x_i

When the domain Δ is clear, we can omit it, and write directly $\langle \mathcal{D}, \mathcal{V} \rangle \models \varphi$, instead of $\langle \langle \Delta, \mathcal{D} \rangle, \mathcal{V} \rangle \models \varphi$

Result of relational calculus queries

Definition

The result of the evaluation of a relational calculus query $q = \{(x_1, \ldots, x_n) \mid \varphi\}$ on a database $\mathcal{D}$ over $\mathcal{A}$ and Δ is the relation $q^{\mathcal{D}}$ such that

- the arity of $q^{\mathcal{D}}$ is n
- the extension of $q^{\mathcal{D}}$ is the set of constants that constitute the value of the query q in the interpretation $\langle \Delta, \mathcal{D} \rangle$

Conjunctive queries

- are the most common kind of relational calculus queries
- also known as select-project-join SQL queries
- allow for easy optimization in relational DBMSs

Definition

A conjunctive query (CQ) is a relational calculus query of the form

$$\{ (\vec{x}) \mid \exists \vec{y}.\, r_1(\vec{x}_1, \vec{y}_1) \wedge \cdots \wedge r_m(\vec{x}_m, \vec{y}_m) \}$$

where

- $\vec{x}$ is the union of the $\vec{x}_i$'s, and $\vec{y}$ is the union of the $\vec{y}_i$'s
- $r_1, \ldots, r_m$ are relation symbols (not built-in predicates)

We use the following abbreviation: $\{ (\vec{x}) \mid r_1(\vec{x}_1, \vec{y}_1), \ldots, r_m(\vec{x}_m, \vec{y}_m) \}$

Complexity of relational calculus

We consider the complexity of the recognition problem, i.e., checking whether a tuple of constants is in the answer to a query:

- measured wrt the size of the database $\leadsto$ data complexity
- measured wrt the size of the query and the database $\leadsto$ combined complexity

Complexity of relational calculus

- data complexity: polynomial, actually in LOGSPACE
- combined complexity: PSPACE-complete

Complexity of conjunctive queries

- data complexity: in LOGSPACE
- combined complexity: NP-complete

Queries to a data integration system $\mathcal{I}$

- The domain Δ is fixed, and we do not distinguish an element of Δ from the constant denoting it $\leadsto$ standard names
- Queries to $\mathcal{I}$ are relational calculus queries over the alphabet $\mathcal{A}_\mathcal{G}$ of the global schema
- When "evaluating" q over $\mathcal{I}$, we have to consider that for a given source database $\mathcal{C}$, there may be many global databases $\mathcal{B}$ in $sem^\mathcal{C}(\mathcal{I})$
- We consider those answers to q that hold for all global databases in $sem^\mathcal{C}(\mathcal{I})$

  $\leadsto$ certain answers

Semantics of queries to $\mathcal{I}$

Definition

Given q, $\mathcal{I}$, and $\mathcal{C}$, the set of certain answers to q wrt $\mathcal{I}$ and $\mathcal{C}$ is

$$cert(q, \mathcal{I}, \mathcal{C}) = \{ \ (c_1, \ldots, c_n) \in q^{\mathcal{B}} \mid \text{for all } \mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I}) \ \}$$

- Query answering is logical implication
- Complexity is measured mainly *wrt the size of the source db* $\mathcal{C}$, i.e., we consider data complexity
- We consider the problem of deciding whether $\vec{c} \in cert(q, \mathcal{I}, \mathcal{C})$, for a given $\vec{c}$

Databases with incomplete information, or knowledge bases

- Traditional database: one model of a first-order theory
  Query answering means evaluating a formula in the model

- Database with incomplete information, or knowledge base: set of models (specified, for example, as a restricted first-order theory)
  Query answering means computing the tuples that satisfy the query in all the models in the set

There is a strong connection between query answering in data integration and query answering in databases with incomplete information under constraints (or, query answering in knowledge bases)

Query answering with incomplete information

- [Reiter '84]: relational setting, databases with incomplete information modeled as a first order theory
- [Vardi '86]: relational setting, complexity of reasoning in closed world databases with unknown values
- Several approaches both from the DB and the KR community
- [van der Meyden '98]: survey on logical approaches to incomplete information in databases

The mapping

How is the mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{G}$ specified?

- Are the sources defined in terms of the global schema?
  Approach called source-centric, or local-as-view, or LAV
- Is the global schema defined in terms of the sources?
  Approach called global-schema-centric, or global-as-view, or GAV
- A mixed approach?
  Approach called GLAV

GAV vs. LAV – Example

Global schema:

movie($Title, Year, Director$)
european($Director$)
review($Title, Critique$)

Source 1:

$r_1(Title, Year, Director)$ since 1960, european directors

Source 2:

$r_2(Title, Critique)$ since 1990

Query: Title and critique of movies in 1998

$\{ (t,r) \mid \exists d.\ \text{movie}(t, 1998, d) \land \text{review}(t, r) \}$, abbreviated

$\{ (t,r) \mid \text{movie}(t, 1998, d),\ \text{review}(t, r) \}$

Formalization of GAV

In GAV (with sound sources), the mapping $\mathcal{M}$ is a set of assertions:

$$\phi_{\mathcal{S}} \leadsto g$$

one for each element g in $\mathcal{A}_{\mathcal{G}}$, with $\phi_{\mathcal{S}}$ a query over $\mathcal{S}$ of the arity of g

Given a source db $\mathcal{C}$, a db $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each $g \in \mathcal{G}$:

$$\phi_{\mathcal{S}}^{\mathcal{C}} \subseteq g^{\mathcal{B}}$$

In other words, the assertion means $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow g(\vec{x})$

Given a source database, $\mathcal{M}$ provides direct information about which data satisfy the elements of the global schema

Relations in $\mathcal{G}$ are views, and queries are expressed over the views.
Thus, it seems that we can simply evaluate the query over the data satisfying the global relations (as if we had a single database at hand).

GAV – Example

Global schema: movie($Title, Year, Director$)
european($Director$)
review($Title, Critique$)

GAV: to each relation in the global schema, $\mathcal{M}$ associates a view over the sources:

$$\{\ (t,y,d)\ \mid\ \mathsf{r}_1(t,y,d)\ \} \ \leadsto\ \mathsf{movie}(t,y,d)$$
$$\{\ (d)\ \mid\ \mathsf{r}_1(t,y,d)\ \} \ \leadsto\ \mathsf{european}(d)$$
$$\{\ (t,r)\ \mid\ \mathsf{r}_2(t,r)\ \} \ \leadsto\ \mathsf{review}(t,r)$$

Logical formalization:

$$\forall t,y,d.\ \mathsf{r}_1(t,y,d) \rightarrow \mathsf{movie}(t,y,d)$$
$$\forall d.\ (\exists t,y.\ \mathsf{r}_1(t,y,d)) \rightarrow \mathsf{european}(d)$$
$$\forall t,r.\ \mathsf{r}_2(t,r) \rightarrow \mathsf{review}(t,r)$$

GAV – Example of query processing

The query

$$\{\ (t, r)\ \mid\ \mathsf{movie}(t, 1998, d),\ \ \mathsf{review}(t, r)\ \}$$

is processed by means of unfolding, i.e., by expanding each atom according to its associated definition in $\mathcal{M}$, so as to come up with source relations

In this case:

$$\begin{array}{llll} \{\ (t, r)\ \mid & \mathsf{movie}(t, 1998, d), & \mathsf{review}(t, r)\ \} \\ \text{unfolding} & \downarrow & \downarrow \\ \{\ (t, r)\ \mid & \mathsf{r}_1(t, 1998, d), & \mathsf{r}_2(t, r)\ \} \end{array}$$

GAV – Example of constraints

Global schema containing constraints:

movie($Title, Year, Director$)
european($Director$)
review($Title, Critique$)
european_movie_60s($Title, Year, Director$)

$\forall t, y, d.$ european_movie_60s$(t, y, d) \rightarrow$ movie(t, y, d)
$\forall d. \exists t, y.$ european_movie_60s$(t, y, d) \rightarrow$ european(d)

GAV mappings:

$\{ (t, y, d) \mid r_1(t, y, d) \} \rightsquigarrow$ european_movie_60s(t, y, d)
$\{ (d) \mid r_1(t, y, d) \} \rightsquigarrow$ european(d)
$\{ (t, r) \mid r_2(t, r) \} \rightsquigarrow$ review(t, r)

Formalization of LAV

In LAV (with sound sources), the mapping $\mathcal{M}$ is a set of assertions:

$$s \leadsto \phi_{\mathcal{G}}$$

one for each source element s in $\mathcal{A}_{\mathcal{S}}$, with $\phi_{\mathcal{G}}$ a query over $\mathcal{G}$

Given source db $\mathcal{C}$, a db $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each $s \in \mathcal{S}$:

$$s^{\mathcal{C}} \subseteq \phi_{\mathcal{G}}^{\mathcal{B}}$$

In other words, the assertion means $\forall \vec{x}.\ s(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$

The mapping $\mathcal{M}$ and the source database $\mathcal{C}$ do not provide direct information about which data satisfy the global schema

Sources are views, and we have to answer queries on the basis of the available data in the views

LAV – Example

Global schema: movie($Title$, $Year$, $Director$)
european($Director$)
review($Title$, $Critique$)

LAV: to each source relation, $\mathcal{M}$ associates a view over the global schema:

$$\mathsf{r}_1(t, y, d) \leadsto \{ (t, y, d) \mid \mathsf{movie}(t, y, d),\ \mathsf{european}(d),\ y \geq 1960 \}$$
$$\mathsf{r}_2(t, r) \leadsto \{ (t, r) \mid \mathsf{movie}(t, y, d),\ \mathsf{review}(t, r),\ y \geq 1990 \}$$

The query $\{ (t, r) \mid \mathsf{movie}(t, 1998, d),\ \mathsf{review}(t, r) \}$ is processed by means of an inference mechanism that aims at re-expressing the atoms of the global schema in terms of atoms at the sources.
In this case:

$$\{ (t, r) \mid \mathsf{r}_2(t, r),\ \mathsf{r}_1(t, 1998, d) \}$$

GAV and LAV – Comparison

GAV: (e.g., Carnot, SIMS, Tsimmis, IBIS, Momis, DisAtDis, . . .)

- Quality depends on how well we have compiled the sources into the global schema through the mapping
- Whenever a source changes or a new one is added, the global schema needs to be reconsidered
- Query processing can be based on some sort of unfolding (query answering looks easier – without constraints)

LAV: (e.g., Information Manifold, DWQ, Picsel)

- Quality depends on how well we have characterized the sources
- High modularity and extensibility (if the global schema is well designed, when a source changes, only its definition is affected)
- Query processing needs reasoning (query answering complex)

Beyond GAV and LAV: GLAV

In GLAV (with sound sources), the mapping $\mathcal{M}$ is a set of assertions:

$$\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$$

with $\phi_{\mathcal{S}}$ a query over $\mathcal{S}$, and $\phi_{\mathcal{G}}$ a query over $\mathcal{G}$ of the same arity as $\phi_{\mathcal{S}}$

Given source db $\mathcal{C}$, a db $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$ in $\mathcal{M}$:

$$\phi_{\mathcal{S}}^{\mathcal{C}} \subseteq \phi_{\mathcal{G}}^{\mathcal{B}}$$

In other words, the assertion means $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$

As for LAV, the mapping $\mathcal{M}$ does not provide direct information about which data satisfy the global schema

To answer a query q over $\mathcal{G}$, we have to infer how to use $\mathcal{M}$ in order to access the source database $\mathcal{C}$

GLAV – Example

Global schema: $\text{work}(Person, Project)$, $\text{area}(Project, Field)$

Source 1: $\text{hasjob}(Person, Field)$
Source 2: $\text{teaches}(Professor, Course)$, $\text{in}(Course, Field)$
Source 3: $\text{get}(Researcher, Grant)$, $\text{for}(Grant, Project)$

GLAV mapping:

$$\{(r, f) \mid \text{hasjob}(r, f)\} \rightsquigarrow \{(r, f) \mid \text{work}(r, p),\ \text{area}(p, f)\}$$
$$\{(r, f) \mid \text{teaches}(r, c),\ \text{in}(c, f)\} \rightsquigarrow \{(r, f) \mid \text{work}(r, p),\ \text{area}(p, f)\}$$
$$\{(r, p) \mid \text{get}(r, g),\ \text{for}(g, p)\} \rightsquigarrow \{(r, f) \mid \text{work}(r, p)\}$$

GLAV – A technical observation

In GLAV (with sound sources), the mapping $\mathcal{M}$ is constituted by a set of assertions:

$$\phi_{\mathcal{S}} \leadsto \phi_{\mathcal{G}}$$

Each such assertion can be rewritten wlog by introducing a new predicate r (not to be used in the queries) of the same arity as the two queries and replace the assertion with the following two:

$$\phi_{\mathcal{S}} \leadsto r \qquad\qquad r \leadsto \phi_{\mathcal{G}}$$

In other words, we replace $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$
with $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow r(\vec{x})$ and $\forall \vec{x}.\ r(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$