View-based query processing

Diego Calvanese
Faculty of Computer Science
Free University of Bolzano/Bozen

Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"

Georg Gottlob
Technische Universitat Wien, Vienna, Austria

Corso di dottorato – Dottorato in Ingegneria Informatica,
Università di Roma "La Sapienza", settembre – ottobre 2005

Schedule: Lectures 1-10

1. Lecture 1,2 - Sept 5, 2005 - hr 14:00 [Lenzerini]
   Introduction to view-based query processing
2. Lecture 3,4 - Sept 9, 2005 - hr 10:30 [Gottlob]
   Conjunctive query evaluation
3. Lecture 5,6 - Sept 9, 2005 - hr 14:00 [Gottlob]
   Data exchange
4. Lecture 7,8 - Sept 19, 2005 - hr 14:00 [De Giacomo]
   Data integration 1
5. Lecture 9,10 - Sept 21, 2005 - hr 14:00 [Rosati]
   Data integration 2

Schedule: Lectures 11-20

6. Lecture 11,12 - Sept 23, 2005 - hr 14:00 [Rosati]
   Data integration 3
7. Lecture 13,14 - Sept 26, 2005 - hr 14:00 [De Giacomo]
   Data integration through ontologies
8. Lecture 15,16 - Oct 13, 2005 - hr 14:00 [Calvanese]
   View-based query processing over semistructured data 1
9. Lecture 17,18 - Oct 14, 2005 - hr 14:00 [Calvanese]
   View-based query processing over semistructured data 2
10. Lecture 19,20 - Oct 17, 2005 - hr 14:00 [Lenzerini]
    Reasoning about views

Lectures 1-2: Outline

1. What is “view-based query processing”
2. Prerequisites for the course
3. Formalization of view-based query processing
4. Applications of view-based query processing
5. Outline of the rest of the course

Views

- A view is a pre-defined query
- In a database management system, a view is defined at the schema level, and then used in the system in several ways (e.g., in queries)
- When processing a query referring to a views, the “unfolding” technique is generally adopted
- Problems: view update, optimization, etc.

What is “view based query processing”

- View based query processing addresses the issue of processing a query by relying solely on a set of views, rather than the raw data

- Relevant problem in
  - database management,
  - data integration,
  - data exchange,
  - data warehousing,
  - access control,
  - mobile computing,
  - knowledge representation,
  - the semantic web

View based query processing

The problem is characterized by several parameters:

1. Data model for expressing the schema
2. Integrity constraints in the schema
3. Language for view definition
4. Assumption on view definition
   - sound, complete, or exact
   - materialized or virtual
5. Assumption on domain
   - open or closed
   - finite or unrestricted
6. Languages for expressing queries
7. What does processing mean (answering, rewriting, reasoning, etc.)

Example of "view based query processing"

Consider the following view definition:

- $v_1(X) \ :- \ p(X,Y)$
- $v_2(Y) \ :- \ p(X,Y)$

and assume that the view instance consists of $\{v_1(a), v_2(b)\}$.

Under the sound view assumption (open world assumption), we only know that some p tuple has a in its first component, and some p tuple has b in its second component.

Under the exact view assumption (closed world assumption) we can conclude that all p tuples have a in their first component and b as their second component, i.e. p contains exactly the tuple (a,b).

What does “processing” mean?

- View-based query answering
- View based query rewriting
- View materialization
- Reasoning on queries and views
  - Query containment (view subsumption)
  - View-based query containment
  - View-losslessness
  - Perfectness/exactness of rewriting

Query languages

- Relational data
  - Relational algebra, relational calculus, (basic) SQL (no ordering, aggregates, etc.), First Order Logic (FOL)
  - Subsets of FOL (conjunctive queries, union of conjunctive queries)
  - Datalog and its variants
- Semi-structured data
  - Regular path queries
  - Extensions to regular path queries
  - Datalog and its variants

Query evaluation over a database

The database B is a finite FOL structure, the query q is a formula, and we want to compute the answers to q over B

$$\{ \vec{\mathbf{t}} \mid B \models q(\vec{\mathbf{t}}), \text{ i.e., } \vec{\mathbf{t}} \in q(B) \}$$

Complexity

- combined complexity - complexity of the following problem: given a database B, a query q, and a tuple $\vec{\mathbf{t}}$, check whether $\vec{\mathbf{t}}$ is an answer to q over B.
- data complexity - for a fixed q, complexity of the following problem: given a database B, and a tuple $\vec{\mathbf{t}}$, check whether $\vec{\mathbf{t}}$ is an answer to q over B.

Query evaluation over a set of databases

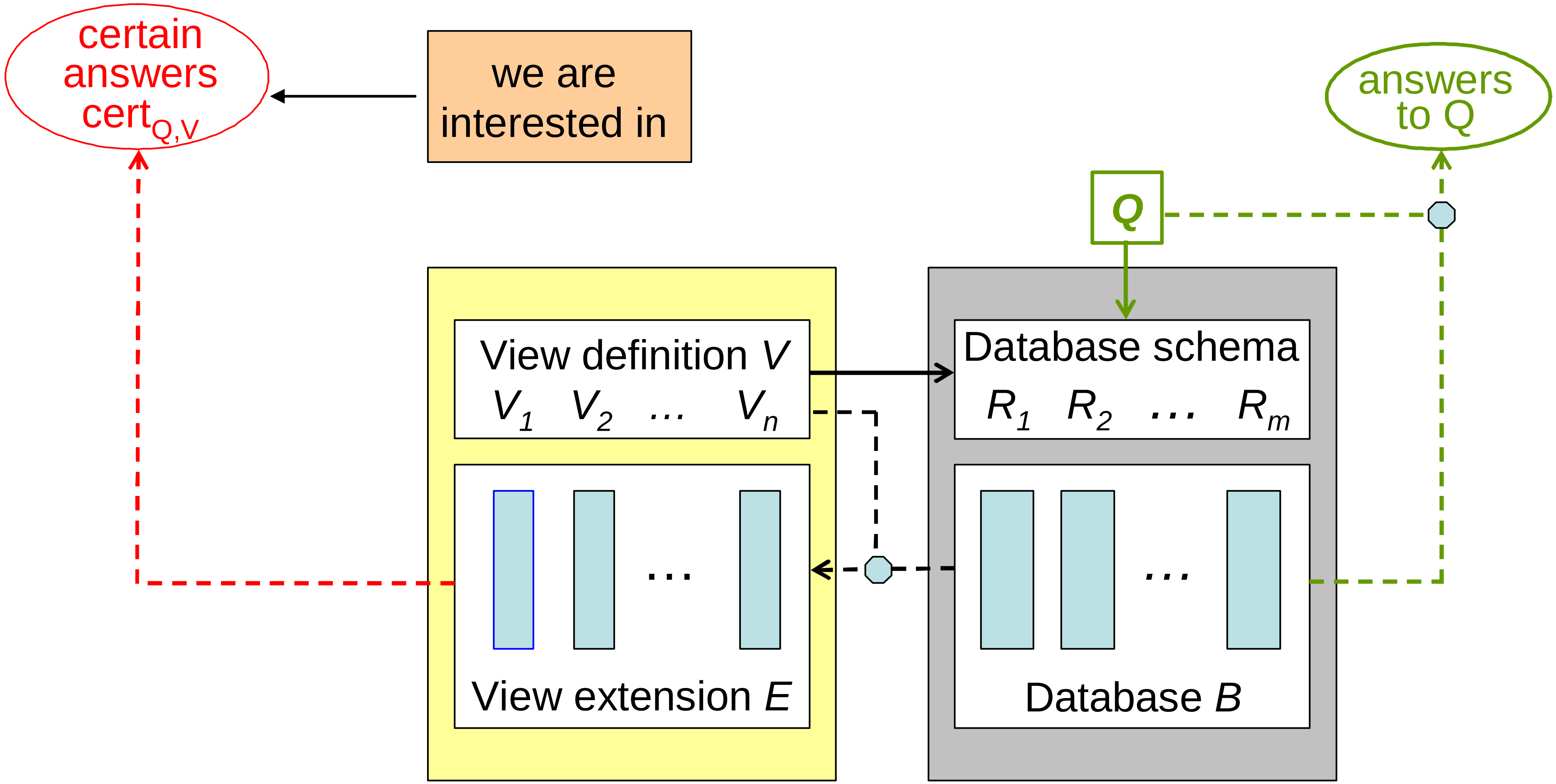
Let Σ be a specification for a set σ of databases (finite or not), constituted by two parts, Σ_i and Σ_e, called intensional and extensional, respectively. The query q is again a formula, and we want to compute the set of certain answers to q over σ (or, over Σ)

$$\{ \vec{\mathbf{t}} \mid \forall B \in \sigma,\ \vec{\mathbf{t}} \in q(B) \}$$

Complexity

- combined complexity - complexity of the following problem: given a specification Σ for a set σ of databases, a query q, and a tuple $\vec{\mathbf{t}}$, check whether $\vec{\mathbf{t}}$ is a certain answer to q over Σ.
- data complexity - for a fixed query q and Σ_i, complexity of the following problem: given the extensional component Σ_e of a specification Σ for a set σ of databases, and a tuple $\vec{\mathbf{t}}$, check whether $\vec{\mathbf{t}}$ is a certain answer to q over Σ.

The main problem: View based query answering



Formalization of view based query answering

Given a schema Σ, a view over Σ is specified by

- one view symbol V and
- one view definition V^Σ, that is a query over Σ

An extension E for view V is a set of tuples (of the same arity as V^Σ).

Given a set $\mathcal{V}$ of views $\{V_1, \ldots, V_n\}$ over Σ, a $\mathcal{V}$-extension $\mathcal{E}$ is a FOL structure over $\{V_1, \ldots, V_n\}$, i.e., a collection $\{E_1, \ldots, E_n\}$ constituted by one extension E_i for each view V_i in $\mathcal{V}$. If V_i is a view in $\mathcal{V}$ and $\mathcal{E} = \{E_1, \ldots, E_n\}$ a $\mathcal{V}$-extension, we write $V_i(\mathcal{E})$ to denote E_i.

Formalization of view based query answering

Given a set $\mathcal{V}$ of views and a database B, we use $\mathcal{V}^{\Sigma}(B)$ to denote the $\mathcal{V}$-extension $\{E_1, \ldots, E_n\}$ such that $V(E_i) = V_i^{\Sigma}(B)$, for each $V_i \in \mathcal{V}$.

We say that a $\mathcal{V}$-extension $\mathcal{E}$ is sound wrt a database B if $\mathcal{E} \subseteq \mathcal{V}^{\Sigma}(B)$, i.e., if $V(\mathcal{E}) \subseteq V^{\Sigma}(B)$ for each $V \in \mathcal{V}$.

In other words, in a $\mathcal{V}$-extension $\mathcal{E}$ that is sound wrt a database B, all the tuples in $V(\mathcal{E})$ appear in $\mathcal{V}^{\Sigma}(B)$, but $\mathcal{V}^{\Sigma}(B)$ may contain tuples not in $V(\mathcal{E})$. Therefore, sound view extensions are extensions that conform to the open world assumption.

In the rest of the course, we always refer to the sound view assumption.

Formalization of view based query answering

A schema Σ, a set $\mathcal{V}$ of views over Σ, a $\mathcal{V}$-extension $\mathcal{E}$, and a domain assumption δ, can be seen as specifying a set of databases, i.e., all databases B that

- satisfy Σ and δ,
- conform to $\mathcal{V}$ and $\mathcal{E}$, i.e., s.t. $\mathcal{V}$-extension $\mathcal{E}$ is sound wrt B.

View-based query answering aims at computing the certain answers of a query wrt such a set of databases: given a schema Σ, a set $\mathcal{V}$ of views over Σ, a $\mathcal{V}$-extension $\mathcal{E}$, and a domain assumption δ, the certain answers (under domain assumption δ) to q with respect to Σ, $\mathcal{V}$ and $\mathcal{E}$ is the set

$$cert_\delta(q, \Sigma, \mathcal{V}, \mathcal{E}) = \{\vec{\mathbf{t}} \mid \vec{\mathbf{t}} \in q(B), \forall B \text{ s.t. } \mathcal{E} \subseteq \mathcal{V}^\Sigma(B) \text{ and } B \text{ satisfies } \delta\}$$

The problem of view based query answering

The decision problem (under a predefined domain assumption δ) is as follows. Given:

- schema Σ,
- set $\mathcal{V}$ of views over Σ,
- $\mathcal{V}$-extension $\mathcal{E}$,
- query q over Σ,
- tuple $\vec{t}$,

check whether $\vec{t} \in cert_\delta(q, \Sigma, \mathcal{V}, \mathcal{E})$.

- combined complexity: wrt the size of all inputs
- data complexity: wrt the size of $\mathcal{E}$ only

Application to access authorization

We have a schema Σ and a finite database B for Σ.

Authorization constraints are modeled by associating to each user U a set $\mathcal{V}_U$ of views, representing the precise collection of data that the user is allowed to know about the database.

Each user may ask queries over Σ to get data from B, but the system should answer the query according the authorization constraints.

Authorization-based access is nicely formalized by view-based query answering: when a user U poses a query q to the database, the systems returns the set $cert_\delta(q, \Sigma, \mathcal{V}_U, \mathcal{V}_U^\Sigma(B))$, where δ is the “open and finite domain assumption”.

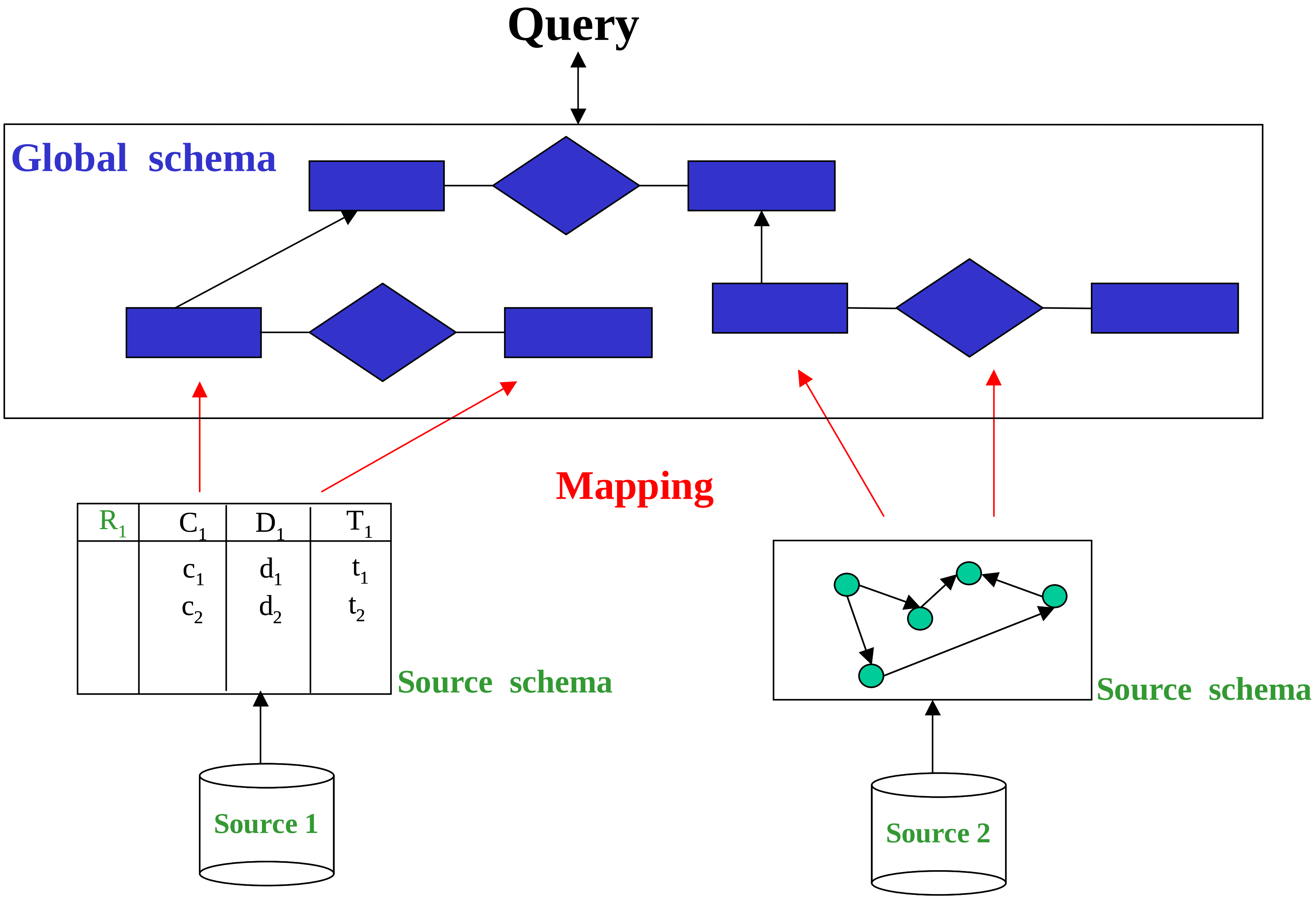
Application to access authorization: example

We have a schema Σ with $jobAddress(x, y)$ (y is the job location of x), and $site(x, y)$ (y is a site of company x), and a database saying that Bob works in SF, and SF is a location of Sony.

Suppose that U is allowed to know who is working for which companies, but is not allowed to know in which addresses a person works, or which are the sites of a company.

We associate $\{ (x, z) \mid \exists y \; jobAddress(x, y) \wedge site(y, z)\}$ to user U, so U gets the empty answer to $jobAddress(''Bob'', z)$, but gets an informative answer to $\{ z \mid \exists y \; jobAddress(''Bob'', y) \wedge site(y, z)\}$.

Application to data integration



Formal framework for data integration

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

- $\mathcal{G}$ is the global schema

  The global schema is a logical theory over an alphabet $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{S}$ is the source schema

  The source schema is constituted simply by an alphabet $\mathcal{A}_{\mathcal{S}}$ disjoint from $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{G}$

  Different approaches to the specification of mapping

Semantics of a data integration system

We refer only to databases over a <u>fixed infinite domain Γ</u> of constants.

Let $\mathcal{C}$ be **a source database** over Γ (also called source model), fixing the extension of the predicates of $\mathcal{A}_{\mathcal{S}}$ (thus modeling the data present in the sources).

The databases that satisfy $\mathcal{I}$ are the logical interpretations for $\mathcal{A}_{\mathcal{G}}$ (called global databases) that satisfy $\mathcal{G}$ under the "open and unrestricted domain assumption" (OU), and satisfy $\mathcal{M}$ wrt $\mathcal{C}$ (what does this mean depends on the nature of the mapping $\mathcal{M}$). By the above definition, $\mathcal{I}$ specifies a set of 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

Example of data integration

Global schema: movie($Title, Year, Director$)

european($Director$)

review($Title, Critique$)

Source 1: r$_1(Title, Year, Director)$ since 1960, euro directors

Source 2: r$_2(Title, Critique)$ since 1990

Query: Title and critique of movies in 1998

$\exists D.$ movie$(T, 1998, D) \wedge$ review(T, R), written

$\{ (T, R) \mid$ movie$(T, 1998, D) \wedge$ review$(T, R) \}$

Semantics of LAV

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

$$s \quad \rightsquigarrow \quad \phi_{\mathcal{G}}$$

one for each source element s in $\mathcal{A}_{\mathcal{S}}$, where $\phi_{\mathcal{G}}$ is a query over $\mathcal{G}$ of the arity of s.

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

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

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

LAV – example

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

LAV: associated to source relations we have views over the global schema

$r_1(T, Y, D) \leadsto \{(T, Y, D) \mid \text{movie}(T, Y, D) \wedge \text{european}(D) \wedge Y \geq 1960\}$

$r_2(T, R) \leadsto \{(T, R) \mid \text{movie}(T, Y, D) \wedge \text{review}(T, R) \wedge Y \geq 1990\}$

Formalizing LAV as view-based query answering

Given a LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and a source database $\mathcal{C}$ for $\mathcal{I}$, we define:

- Schema Σ: global schema $\mathcal{G}$ of $\mathcal{I}$
- Views $\mathcal{V}$: one view for each source in $\mathcal{S}$
- View definition for view V: the query that $\mathcal{M}$ associates to source V
- View extension $\mathcal{E}$: source database $\mathcal{C}$

It is easy to see that the answers to a query q posed to $\mathcal{I}$ wrt $\mathcal{C}$ are exactly $cert_{OU}(q, \Sigma, \mathcal{V}, \mathcal{E})$.

Semantics of GAV

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

$$g \quad \rightsquigarrow \quad \phi_{\mathcal{S}}$$

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

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

$$g^{\mathcal{B}} \quad \supseteq \quad {\phi_{\mathcal{S}}}^{\mathcal{C}}$$

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

GAV – example

Global schema: movie$(Title, Year, Director)$

european$(Director)$

review$(Title, Critique)$

GAV: associated to relations in the global schema we have views over the sources

$$\text{movie}(T, Y, D) \leadsto \{ (T, Y, D) \mid \text{r}_1(T, Y, D) \}$$

$$\text{european}(D) \leadsto \{ (D) \mid \text{r}_1(T, Y, D) \}$$

$$\text{review}(T, R) \leadsto \{ (T, R) \mid \text{r}_2(T, R) \}$$

Formalizing GAV as view-based query answering

Given a GAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and a source database $\mathcal{C}$ for $\mathcal{I}$, we define:

- Schema Σ: global schema $\mathcal{G}$ of $\mathcal{I}$
- Views $\mathcal{V}$: one view V' for each symbol V in $\mathcal{G}$ comparing in the mapping $\mathcal{M}$
- View definition for view V': simply V
- View extension $\mathcal{E}$: for each V', the extension of V' is the result of evaluating the query that $\mathcal{M}$ associates to V over $\mathcal{C}$

It is easy to see that the answers to a query q posed to $\mathcal{I}$ wrt $\mathcal{C}$ are exactly $cert_{OU}(q, \Sigma, \mathcal{V}, E)$.

Beyond GAV and LAV: GLAV

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

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

where $\phi_{\mathcal{S}}$ is a query over $\mathcal{S}$, and $\phi_{\mathcal{G}}$ is a query over $\mathcal{G}$ of the arity $\phi_{\mathcal{S}}$.

Given source database $\mathcal{C}$, a database $\mathcal{B}$ that is legal wrt $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each assertion in $\mathcal{M}$:

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

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

Example of GLAV

Global schema: $Work(Person, Project)$, $Area(Project, Field)$

Source 1: $HasJob(Person, Field)$

Source 2: $Teach(Professor, Course)$, $In(Course, Field)$

Source 3: $Get(Researcher, Grant)$, $For(Grant, Project)$

GLAV mapping:

$$\{ (r, f) \mid HasJob(r, f) \} \rightsquigarrow \{ (r, f) \mid Work(r, p) \land Area(p, f) \}$$

$$\{ (r, f) \mid Teach(r, c) \land In(c, f) \} \rightsquigarrow \{ (r, f) \mid Work(r, p) \land Area(p, f) \}$$

$$\{ (r, p) \mid Get(r, g) \land For(g, p) \} \rightsquigarrow \{ (r, p) \mid Work(r, p) \}$$

Formalizing GLAV as view-based query answering

Given a GLAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and a source database $\mathcal{C}$ for $\mathcal{I}$, we define:

- Schema Σ: global schema $\mathcal{G}$ of $\mathcal{I}$
- Views $\mathcal{V}$: one view m for each mapping assertion m in $\mathcal{M}$
- View definition for view m: the query over $\mathcal{G}$ contained in m
- View extension $\mathcal{E}$: for each m, the extension of m is the result of evaluating the query over $\mathcal{S}$ contained in m over $\mathcal{C}$

It is easy to see that the answers to a query q posed to $\mathcal{I}$ wrt $\mathcal{C}$ are exactly $cert_{OU}(q, \Sigma, \mathcal{V}, \mathcal{E})$.

Application to data exchange

The data exchange problem can be informally described as follows.

We have a source S characterized by a schema G_S and a finite database B_S, a target characterized by a schema G_T, and a mapping from G_S to G_T.

The problem is to transfer data from the source to the target according to the mapping. More precisely, we want to materialize in the target a finite database B_T that satisfies G_T and that reflects at best the data coming from the source through the mapping.

Formalizing data exchange as view-based query answering

Given a data exchange setting with source S, target T and mapping M, we define:

- Schema Σ: schema G_T
- Views $\mathcal{V}$: one view m for each mapping assertion m in M
- View definition for view m: the query over G_T contained in m
- View extension $\mathcal{E}$: for each m, the extension of m is the result of evaluating the query over G_S contained in m over B_S

It is easy to see that a finite database B_T reflects at best the data coming from the source through the mapping if for all query q over G_T, $q(B_T) = cert_F(q, \Sigma, \mathcal{V}, \mathcal{E})$, where F stands for the "finite domain assumption".

Outline of the rest of the course

- Lecture 3,4: Conjunctive query evaluation
- Lecture 5,6: Data exchange
- Lecture 7,8: Data integration 1
- Lecture 9,10: Data integration 2
- Lecture 11,12: Data integration 3
- Lecture 13,14: Data integration through ontologies
- Lecture 15,16: View-based query processing over semistructured data 1
- Lecture 17,18: View-based query processing over semistructured data 2
- Lecture 19,20: Reasoning about views