Inconsistency and Incompleteness in Data Integration: a Logic-based Approach

Andrea Calì

Università di Roma “La Sapienza”

CoLogNET Workshop

Logic-based methods for information integration

Vienna, Austria, 23 August 2003

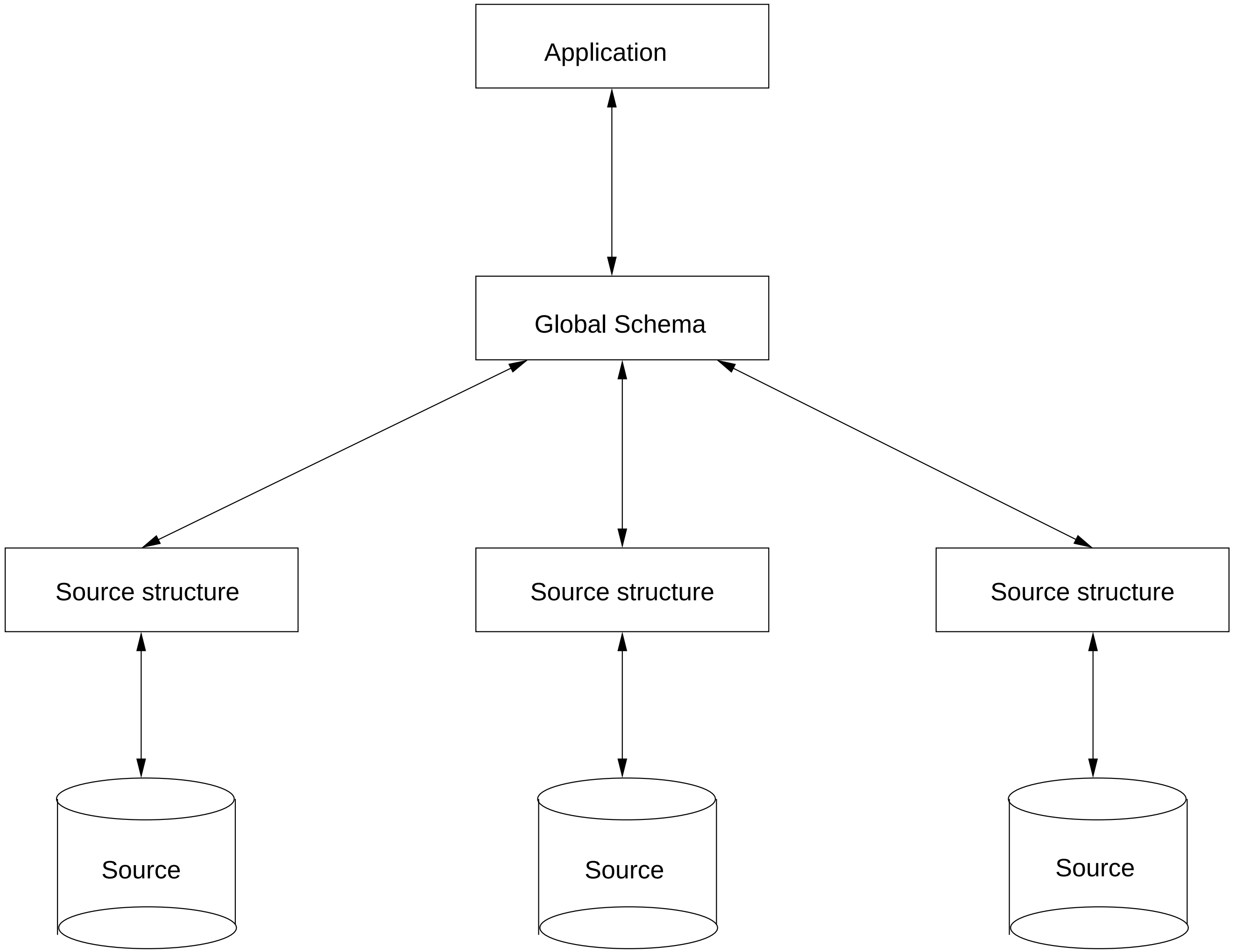
Joint work with:

- Maurizio Lenzerini
- Domenio Lembo
- Riccardo Rosati

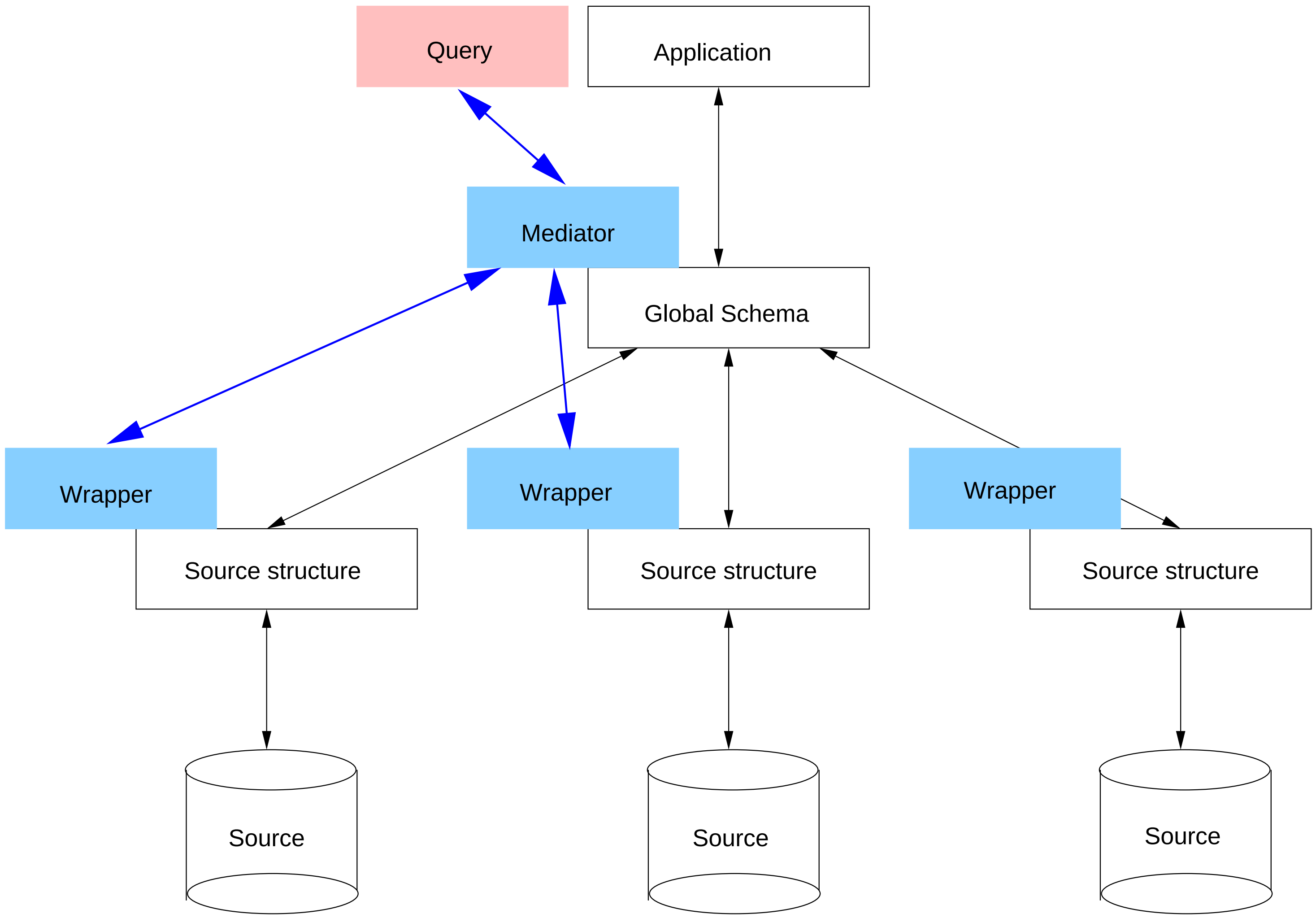
What is a data integration system?

- Offers uniform access to a set of heterogeneous sources
- The representation provided to the user is called **global schema**
- The user is freed from the knowledge about the data sources
- When the user issues a query over the global schema, the system:
  1. determines which sources to query and how
  2. issues suitable queries to the sources
  3. assembles the results and provides the answer to the user

Logical architecture for a data integration system



Software architecture for a data integration system



Data integration system: formalisation

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

- $\mathcal{G}$: global schema
- $\mathcal{S}$: source schema
- $\mathcal{M}$: mapping

Our framework

- **global schema** $\mathcal{G}$: relational with integrity constraints (ICs)
- **source schema** $\mathcal{S}$: relational;
- **mapping** $\mathcal{M}$: **global-as-view (GAV)**, expressed with the language of **union of conjunctive queries (UCQ)**

Example

Global schema: player$(Pname, Pteam)$

team$(Tname, Tcity)$

Source schema: $\{\, s_1/3,\ s_2/2,\ s_3/2 \,\}$

The GAV mapping associates to each relation in the global schema $\mathcal{G}$ a view over the source schema:

$$\text{player} \quad \leadsto \quad \left\{ \begin{array}{lcl} \text{player}(X, Y) & \leftarrow & s_1(X, Y, Z) \\ \text{player}(X, Y) & \leftarrow & s_3(X, Y) \end{array} \right.$$

$$\text{team} \quad \leadsto \quad \text{team}(X, Y) \quad \leftarrow \quad s_2(X, Y)$$

The role of integrity constraints (ICs)

ICs on the global schema:

- enhance the expressiveness of the global schema
- in general **they are not satisfied** by the data at the sources

ICs on the source schema:

- represent local properties of data sources
- we assume that the data at the sources satisfy ICs expressed over the sources

$\Rightarrow$ not considered

Constraints on the global schema

1. key dependencies (KDs)

   $key(r) = \{A_1, \ldots, A_k\}$

2. inclusion dependencies (IDs) (generalisation of foreign key dependencies)

   $r_1[A_1, \ldots, A_m] \subseteq r_2[B_1, \ldots, B_m]$

Outline

- ◇ Introduction (done)
- ◇ Framework (done)
- Reasoning on integrity constraints
- Query rewriting for IDs alone
- Query rewriting for KDs and IDs
- Semantics for inconsistent data (loosely-sound)
- Query rewriting under loosely-sound semantics
- Complexity results

Reasoning about constraints

Given a source database $\mathcal{D}$ for a system $\mathcal{I}$, a global database $\mathcal{B}$ is said to be **legal** if:

1. it satisfies the ICs on the global schema
2. it satisfies the mapping, i.e. $\mathcal{B}$ is constituted by a **superset** of the **retrieved global database** $ret(\mathcal{I}, \mathcal{D})$

- $ret(\mathcal{I}, \mathcal{D})$ is obtained by evaluating, for each relation in $\mathcal{G}$, the mapping queries over the source database
- assumption of **sound mapping**
- there are several global databases that are legal for the system

Answers to queries under constraints

- We are interested in **certain answers**.
- A tuple t is a **certain answer** for a query Q if t is in the answer to Q for **all** (possibly infinite) legal databases.
- The certain answers to Q are denoted by $ans(Q, \mathcal{I}, \mathcal{D})$.

Example

Global schema: $\text{player}(Pname, Pteam)$

$\text{team}(Tname, Tcity)$

Constraints:

$$\text{player}[Pteam] \subseteq \text{team}[Tname]$$

Mapping:

$$\text{player} \rightsquigarrow \begin{cases} \text{player}(X,Y) & \leftarrow \quad s_1(X,Y,Z) \\ \text{player}(X,Y) & \leftarrow \quad s_3(X,Y) \end{cases}$$

$$\text{team} \rightsquigarrow \text{team}(X,Y) \quad \leftarrow \quad s_2(X,Y)$$

Example (cont'd)

Source database $\mathcal{D}$

s_1

figo	$\mathit{realMadrid}$	31

s_2

$\mathit{realMadrid}$	madrid

s_3

totti	roma

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$

team

$realMadrid$	$madrid$

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$

team

$realMadrid$	$madrid$
$roma$	α

The ID on the global schema tells us that $roma$ **is the name of some team**

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain of the database

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$

team

$realMadrid$	$madrid$
$roma$	α

The ID on the global schema tells us that $roma$ is the name of some team

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain of the database

Warning 1 there may be an **infinite number** of legal databases for $\mathcal{I}$

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$

team

$realMadrid$	$madrid$
$roma$	α

The ID on the global schema tells us that $roma$ is the name of some team

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain of the database

Warning 1 there may be an **infinite number** of legal databases for $\mathcal{I}$

Warning 2 in case of cyclic IDs, legal databases for $\mathcal{I}$ may be of **infinite size**

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$

team

$realMadrid$	$madrid$
$roma$	α

The ID on the global schema tells us that $roma$ **is the name of some team**

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain of the database

Consider the query

$$q(X) \leftarrow \mathsf{team}(X, Y)$$

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$

team

$realMadrid$	$madrid$
$roma$	α

The ID on the global schema tells us that $roma$ **is the name of some team**

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain of the database

Consider the query

$$q(X) \leftarrow \text{team}(X, Y)$$

$$ans(q, \mathcal{I}, \mathcal{D}) = \{realMadrid, roma\}$$

Query rewriting

Given a user query Q over $\mathcal{G}$

- we look for a rewriting R of Q expressed over $\mathcal{S}$
- a rewriting R is **perfect** if $R^{\mathcal{D}} = ans(Q, \mathcal{I}, \mathcal{D})$ for every source database $\mathcal{D}$.

With a perfect rewriting, we can do **query answering by rewriting**

Note that we avoid the construction of the retrieved global database $ret(\mathcal{I}, \mathcal{D})$

Query rewriting for IDs alone

Intuition: Use the IDs as basic rewriting rules

$$q(X) \leftarrow \mathsf{team}(X, Y)$$

$$\mathsf{player}[Pteam] \subseteq \mathsf{team}[Tname]$$

as a logic rule: $\mathsf{team}(W_2, W_3) \leftarrow \mathsf{player}(W_1, W_2)$

Query rewriting for IDs alone

Intuition: Use the IDs as basic rewriting rules

$$q(X) \leftarrow \mathsf{team}(X, Y)$$

$$\mathsf{player}[Pteam] \subseteq \mathsf{team}[Tname]$$

as a logic rule: $\mathsf{team}(W_2, W_3) \leftarrow \mathsf{player}(W_1.W_2)$

Basic rewriting step:

when the atom unifies with the **head** of the rule

substitute the atom with the **body** of the rule

We add to the rewriting the query

$$q(X) \leftarrow \mathsf{player}(W, X)$$

Query Rewriting for IDs alone: algorithm ID-rewrite

Iterative execution of:

1. **reduction:** atoms that are subsumed by another atom are eliminated
2. **basic rewriting step**

Main result

- $\Pi_{\mathcal{M}}$: rules of the mapping
- Π_{ID}: union of CQs produced by the rewriting algorithm

Theorem: $\Pi_{\mathcal{M}} \cup \Pi_{ID}$ is a perfect rewriting of the user query Q

Query answering under IDs and KDs

- possibility of inconsistencies (recall the **sound** mapping)
- when $ret(\mathcal{I}, \mathcal{D})$ violates the KDs, no legal database exists and query answering becomes trivial!

Theorem: Query answering under IDs and KDs is undecidable.

Proof: by reduction from implication of IDs and KDs.

Separation

Non-key-conflicting IDs (NKCIDs) are of the form

$$r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$$

where either:

1. no KD is defined over r_2
2. $\mathbf{A}_2$ is **not** a strict superset of $key(r_2)$

Theorem (separation): Under KDs and NKCIDs, when $ret(\mathcal{I}, \mathcal{D})$ satisfies the KDs, KDs can be ignored wrt certain answers

Query rewriting under IDs and NKCIDs

Set of rules Π_{KD} for considering the case of KD violation for a relation r:

$$\begin{array}{rcl} q(Y_1,\ldots,Y_n) & \leftarrow & r(X_1,\ldots,X_k,\ldots,X_i,\ldots), \\ & & r(X_1,\ldots,X_k,\ldots,X_i',\ldots), \\ & & X_i \neq X_i', val(Y_1),\ldots,val(Y_n) \end{array}$$

$X_1,\ldots,X_k$ are the variables corresponding to the attributes of $key(r)$

Theorem: $\Pi_{ID} \cup \Pi_{KD} \cup \Pi_{val} \cup \Pi_{\mathcal{M}}$ is a perfect rewriting of Q.

Π_{val} is the set of rules that imposes that $val(c)$ is true if c is a value in the database.

Semantics for inconsistent data sources

- Under the (strictly) sound semantics, a single KD violation leads to a non-interesting case for query answering
- **New approach: loosely-sound semantics.** Add as much as you like (as with sound semantics), and throw away the minimum number of tuples

A global database $\mathcal{B}_1$ is **better** than another database $\mathcal{B}_2$, denoted $\mathcal{B}_1 \gg_{(\mathcal{I},\mathcal{D})} \mathcal{B}_2$, iff

$$\mathcal{B}_1 \cap ret(\mathcal{I},\mathcal{D}) \supset \mathcal{B}_2 \cap ret(\mathcal{I},\mathcal{D})$$

The answers $ans_\ell(Q,\mathcal{I},\mathcal{D})$ to a query are those that are true on **all** “best” legal global databases w.r.t. $\gg_{(\mathcal{I},\mathcal{D})}$.

Example

Global schema: $\text{player}(Pname, Pteam)$

$\text{team}(Tname, Tcity)$

Constraints:

$$\text{player}[Pteam] \subseteq \text{team}[Tname]$$

$$key(\text{player}) = \{Pname\}$$

Mapping:

$$\text{player} \rightsquigarrow \begin{cases} \text{player}(X, Y) \leftarrow s_1(X, Y, Z) \\ \text{player}(X, Y) \leftarrow s_3(X, Y) \end{cases}$$

$$\text{team} \rightsquigarrow \text{team}(X, Y) \leftarrow s_2(X, Y)$$

Example (cont'd)

Source database $\mathcal{D}$

s_1

$figo$	$realMadrid$	31

s_2

$realMadrid$	$madrid$

s_3

$totti$	$roma$
$figo$	$cavese$

Example (cont'd)

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$

player

$figo$	$realMadrid$
$totti$	$roma$
$figo$	$cavese$

team

$realMadrid$	$madrid$

There are two possible ways of repairing the violation with a minimum deletion of tuples.

Example (cont'd)

First form

player

figo	$\mathit{realMadrid}$
totti	roma

team

$\mathit{realMadrid}$	madrid
roma	α

Example (cont'd)

Second form

player

$totti$	$roma$
$figo$	$cavese$

team

$realMadrid$	$madrid$
$roma$	α
$cavese$	β

For the query

$$q(X) \leftarrow \mathsf{team}(\mathsf{X}, \mathsf{Y})$$

we have

$$ans_{\ell}(q, \mathcal{I}, \mathcal{D}) = \{roma, realMadrid\}$$

Query rewriting under the loosely-sound semantics

Set of rules $\Pi_{\ell KD}$ that take KDs into account (Datalog$^{\neg}$ under stable model semantics): for each relation r in $\mathcal{G}$

$$
\begin{array}{lcl}
r(\mathbf{x}, \mathbf{y}) & \leftarrow & r_{\mathcal{D}}(\mathbf{x}, \mathbf{y}) \ , \ not \ \overline{r}(\mathbf{x}, \mathbf{y}) \\
\overline{r}(\mathbf{x}, \mathbf{y}) & \leftarrow & r_{\mathcal{D}}(\mathbf{x}, \mathbf{y}) \ , \ r(\mathbf{x}, \mathbf{z}) \ , \ Y_1 \neq Z_1 \\
 & \cdots & \\
\overline{r}(\mathbf{x}, \mathbf{y}) & \leftarrow & r_{\mathcal{D}}(\mathbf{x}, \mathbf{y}) \ , \ r(\mathbf{x}, \mathbf{z}) \ , \ Y_m \neq Z_m
\end{array}
$$

where: in $r(\mathbf{x}, \mathbf{y})$ the variables in $\mathbf{x}$ correspond to the attributes constituting the key of the relation r; $\mathbf{y} = Y_1, \ldots, Y_m$ and $\mathbf{z} = Z_1, \ldots, Z_m$.

Query rewriting under the loosely-sound semantics (cont'd)

$\Pi_{\mathcal{MD}}$: rules obtained from $\Pi_{\mathcal{M}}$ by replacing each r with $r_{\mathcal{D}}$.

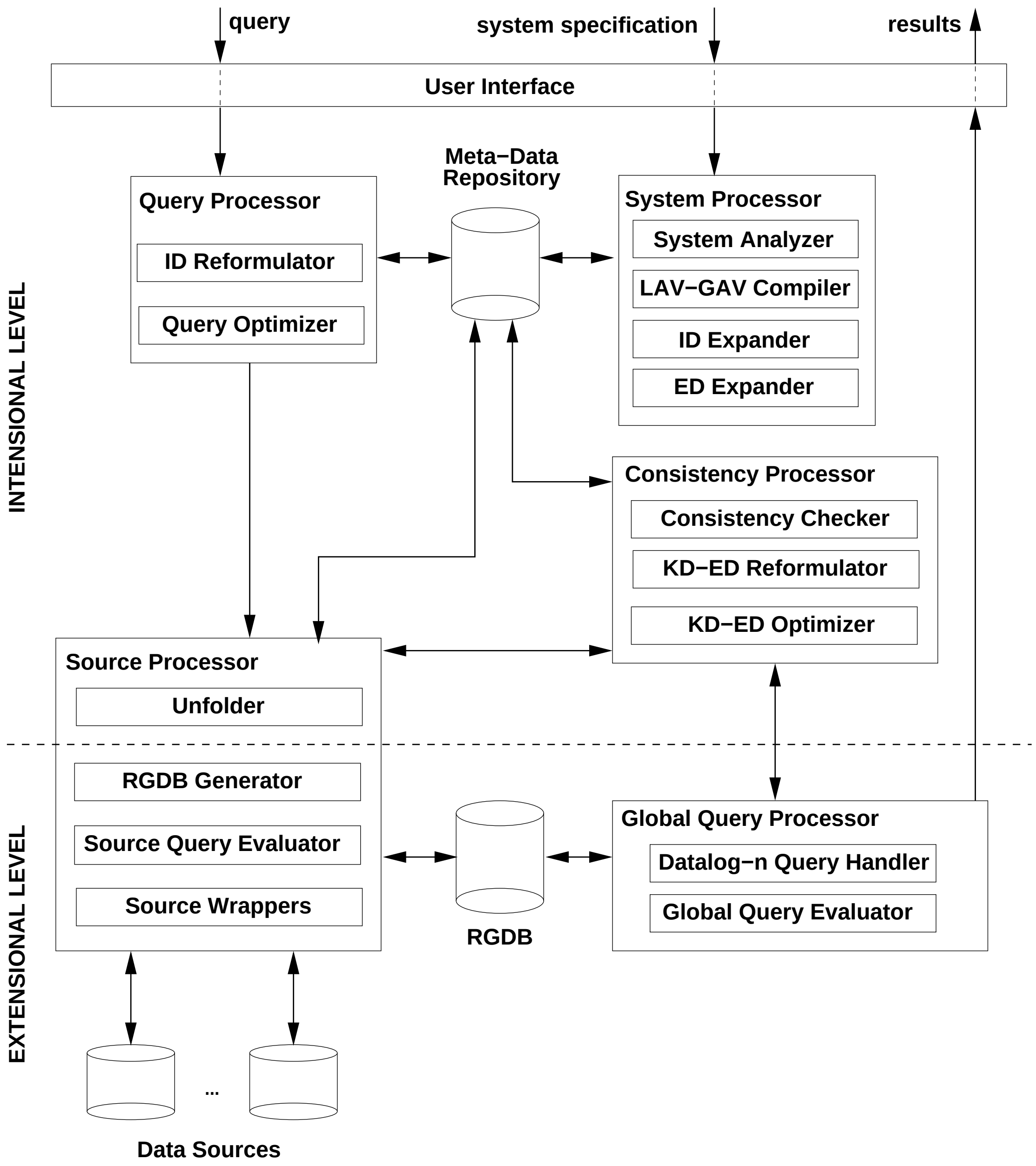
Theorem: $\Pi_{\ell KD} \cup \Pi_{ID} \cup \Pi_{\mathcal{MD}}$ is a perfect rewriting of Q.

Summary of complexity results

KDs	IDs	strictly-sound	loosely-sound
no	GEN	PTIME/PSPACE$^\spadesuit$	PTIME/PSPACE$^\spadesuit$
yes	no	PTIME/NP$^\spadesuit$	coNP/$\Pi_2^p$$^\spadesuit$
yes	FK	PTIME/PSPACE	coNP/PSPACE
yes	NKC	PTIME/PSPACE	coNP/PSPACE
yes	1KC	undecidable	undecidable
yes	GEN	undecidable$^\spadesuit$	undecidable$^\spadesuit$

The system DIS@DIS

- Deals also with **exclusion dependencies**
- The rules $\Pi_{\mathcal{M}}$ are taken into account by means of **unfolding** (substitution)
- Is able to work with **local-as-view (LAV)** mappings, which are translated into GAV ones (plus integrity constraints) [— ER 2002]


query
system specification
results
User Interface
Meta−Data Repository
Query Processor
ID Reformulator
Query Optimizer
System Processor
System Analyzer
LAV−GAV Compiler
ID Expander
ED Expander
Consistency Processor
Consistency Checker
KD−ED Reformulator
KD−ED Optimizer
INTENSIONAL LEVEL
Source Processor
Unfolder
RGDB Generator
Source Query Evaluator
Source Wrappers
RGDB
Global Query Processor
Datalog−n Query Handler
Global Query Evaluator
EXTENSIONAL LEVEL
...
Data Sources

Conclusions

- Query answering by rewriting in data integration systems under constraints
- Query rewriting technique for IDs alone, inspired by [Gryz ICDE 1999]
- Characterisation of the threshold between decidability and undecidability under KDs and IDs [— PODS 2003]
- Query rewriting technique for a maximal class of KDs and IDs
- Loose semantics under KDs and IDs
  - ⋆ Query rewriting technique for KDs, decoupled from that for IDs
- **All rewritings in purely intensional fashion**
- System DIS@DIS

Thank you

Further information:

- Questions: now
- Presenter's contact: `http://www.andreacali.com`
- Implementation: system **DIS@DIS** — try it online! available from the link above

Thanks to: Maurizio Lenzerini, Giuseppe De Giacomo, Diego Calvanese, Jarek Gryz

Slides typesetted with LaTeX2e

Algorithm ID-rewrite

Input: relational schema Ψ, set of IDs Σ_I, UCQ Q

Output: perfect rewriting of Q

$Q' := Q$;

repeat

  $Q_{aux} := Q'$;

  for each $q \in Q_{aux}$ **do**

  (a) **for each** $g_1, g_2 \in body(q)$ **do**

    if g_1 and g_2 unify

    then $Q' := Q' \cup \{\tau(\textit{reduce}(q, g_1, g_2))\}$;

  (b) **for each** $g \in body(q)$ **do**

    for each $I \in \Sigma_I$ **do**

      if I is applicable to g

      then $Q' := Q' \cup \{\, q[g/\textit{gr}(g, I)]\, \}$

until $Q_{aux} = Q'$;

return Q'