Ontology-Based Data Access and Integration

Diego Calvanese

Free University of Bozen-Bolzano

Dept. Enginyeria de Serveis i Sistemes d'Informació
Univ. Politecnica de Catalunya, Barcelona, Spain
February 8th, 2010

Overview

unibz.it

Outline

unibz.it

Data integration

Data integration is the problem of providing a unified and transparent access to a set of autonomous and heterogeneous data sources.

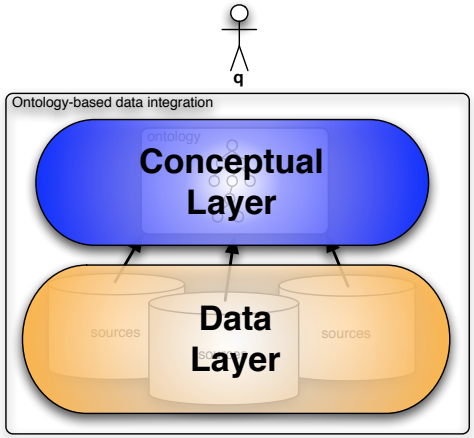
From [Bernstein & Haas, CACM Sept. 2008]:

- Large enterprises spend a great deal of time and money on information integration (e.g., 40% of information-technology shops' budget).
- Market for data integration software estimated to grow from \$2.5 billion in 2007 to \$3.8 billion in 2012 (+8.7% per year) [IDC. Worldwide Data Integration and Access Software 2008-2012 Forecast. Doc No. 211636 (Apr. 2008)]
- Information integration is a large and growing part of science, engineering, and biomedical computing.

unibz.it

Approach: Semantic data integration

Semantic data integration is based on the idea of decoupling information access from data storage:

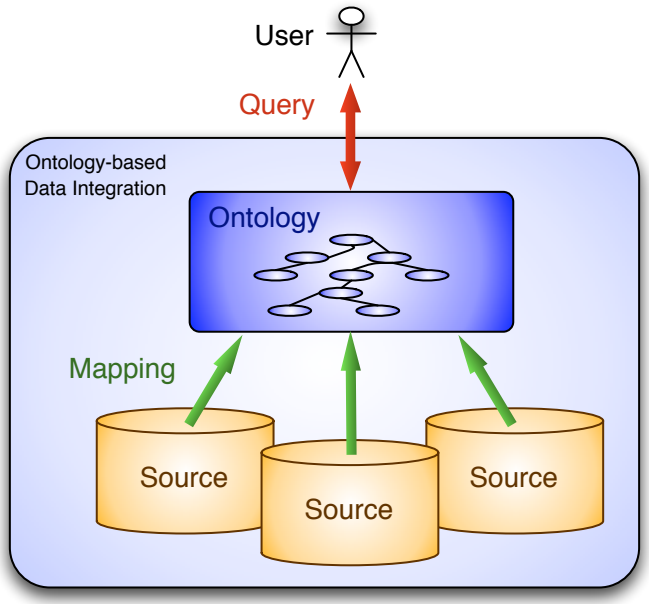


- Clients access only a **conceptual layer** ...
- ... while the **data layer**, hidden to clients, manages the data.

$\rightsquigarrow$ Technological concerns (and changes) on the managed data become fully transparent to the clients.

unibz.it

Semantic data integration: Architecture



Based on three main components:

- **Conceptual layer**: providing a unified view of the data. Is given in terms of an **ontology** $\rightsquigarrow$ Ontology-based data integration.
- **Data sources**: are external, independent, heterogeneous, multiple information systems.
- **Mappings**: semantically link data at the sources with the ontology *(key issue!)*.

Ontologies

An **ontology** is a representation scheme that describes a **formal conceptualization** of a domain of interest.

The specification of an ontology usually comprises:

- An **intensional level**, specifying a set of conceptual elements and of rules to describe the conceptual structures of the domain.
- An **extensional level**, specifying a set of instances of the conceptual elements.

We are concerned here with the intensional level of an ontology.
The role of the extensional level will be played by the data sources.

unibz.it

Intensional level of an ontology

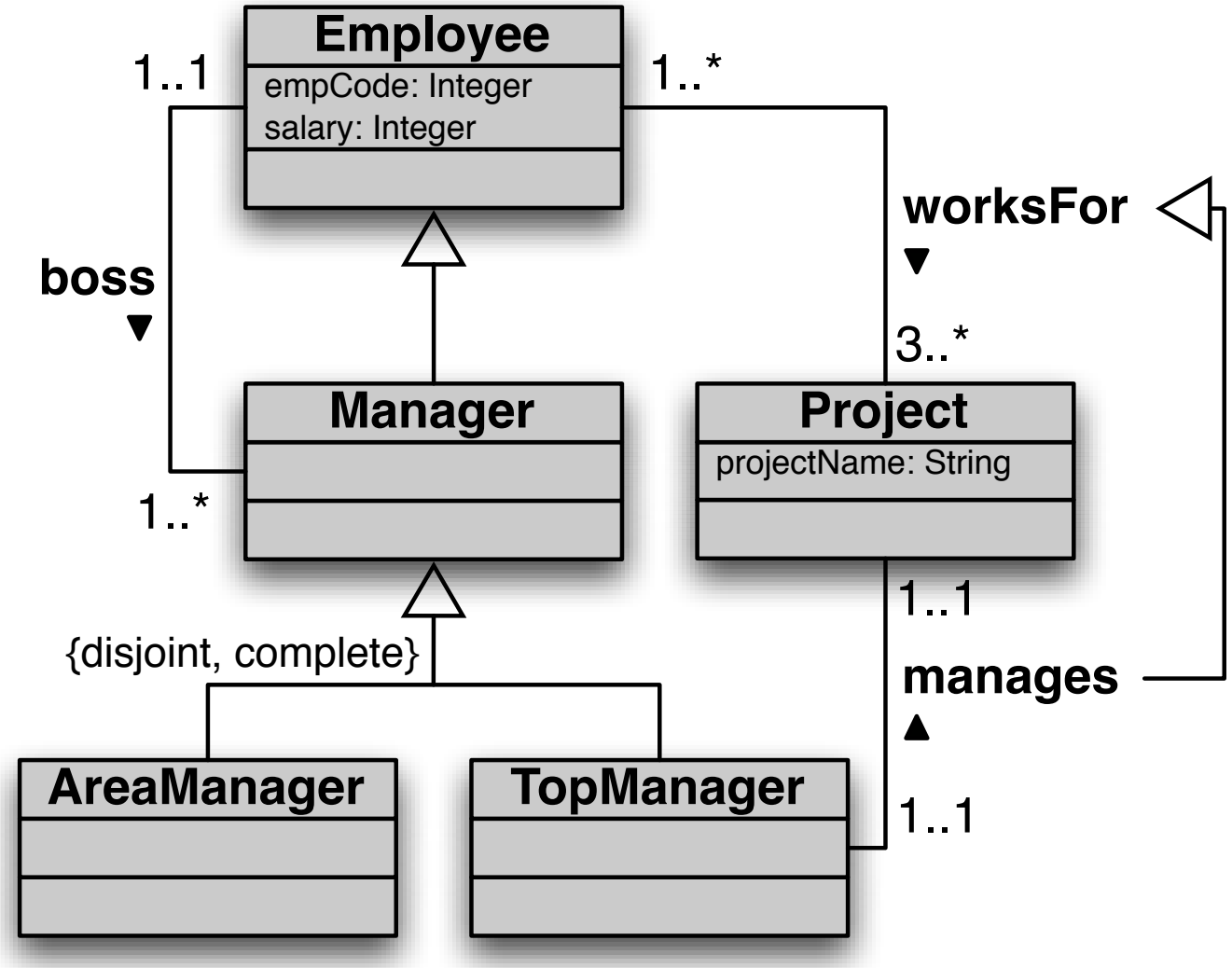
An ontology language for expressing the intensional level usually includes:

- Concepts (or classes), denoting sets of individuals;
- Relationships (or associations, roles) between concepts;
- Properties (or attributes) of concepts and relationships;
- Axioms, capturing additional domain knowledge.

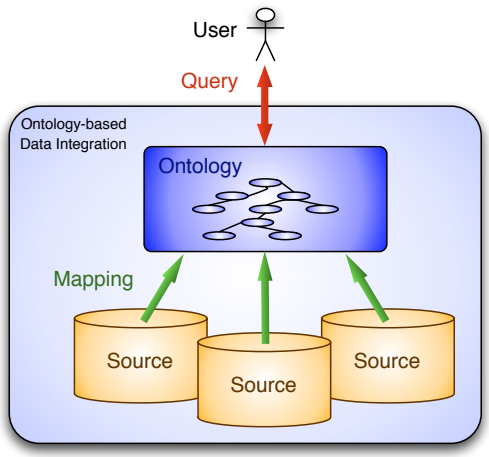
Ontologies are **formalized in logic**, but often **rendered as diagrams** (e.g., Semantic Networks, Entity-Relationship schemas, UML Class Diagrams).

unibz.it

Example: ontology rendered as UML Class Diagram



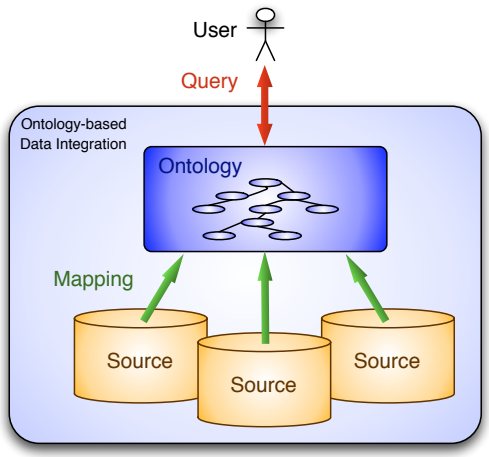
Ontology-based data integration: Conceptual layer



The ontology is used as the conceptual layer, to give clients a unified conceptual global view of the data.

Note: in standard information systems, UML Class Diagram or ER is used at **design time**, . . .
. . . here we use ontologies at **runtime**!

Ontology-based data integration: Sources



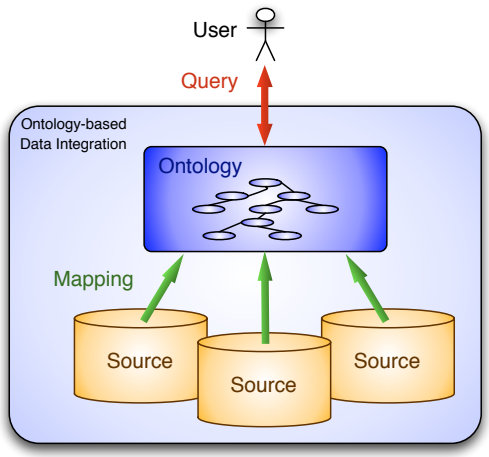
Data sources are external, independent, heterogeneous, multiple information systems.

By now we have industrial solutions for:

- Distributed database systems
- Distributed query optimization
- Tools for source wrapping
- Systems for database federation, e.g., IBM Information Integrator, (but no open-source federated databases yet!)

unibz.it

Ontology-based data integration: Sources



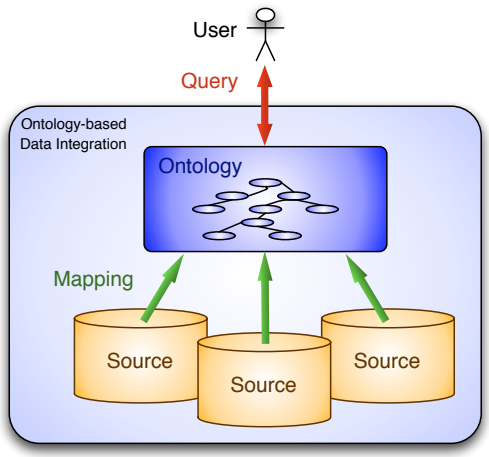
Data sources are external, independent, heterogeneous, multiple information systems.

Based on these industrial solutions we can:

1. Wrap the sources and see all of them as relational databases.
2. Use federated database tools to see the multiple sources as a single one.

$\rightsquigarrow$ We can see the sources as a single (remote) relational database.

Ontology-based data integration: Mappings

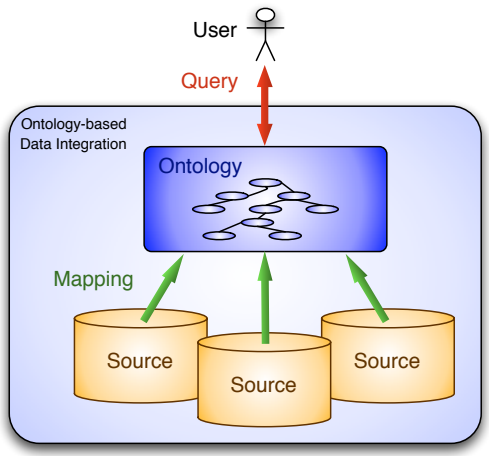


Mappings semantically link data at the sources with the ontology.

Scientific literature on data integration in databases has shown that ...
... in general we cannot simply **map** single relations to single elements of the global view (the ontology).

$\rightsquigarrow$ We need to rely on **queries**!

Ontology-based data integration: Mappings



Mappings semantically link data at the sources with the ontology.

Several forms of mappings based on queries have been considered:

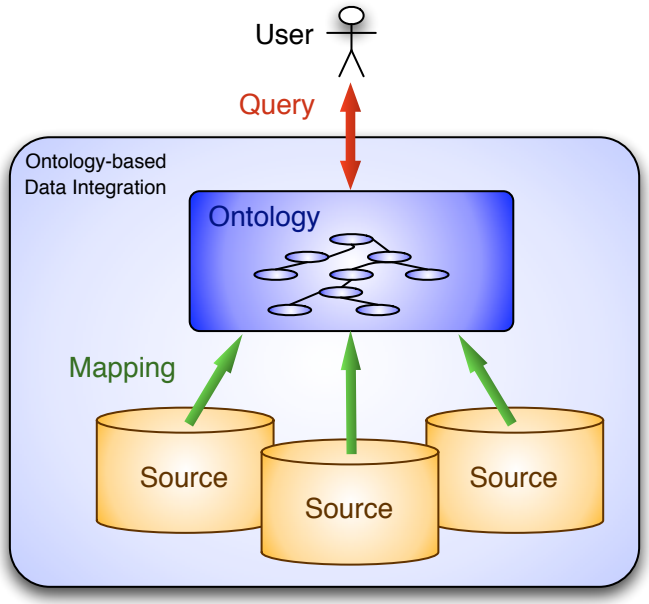
- GAV: map each element in the global view to a query over the sources
  – *most used form of mappings*
- LAV: map each relation in the sources to a query over the global view
  – *mathematically elegant, but practically useless (data not clean enough!)*
- GLAV: map a query over the sources to a query over the global view
  – *generalize both LAV and GAV mappings*

This is a key issue (more on this later).

unibz.it

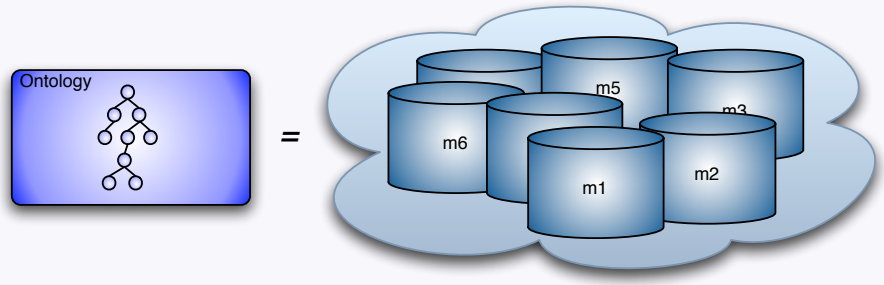
Ontology-based data integration: Incomplete information

Even in standard data integration, the information that the global view has on the data is assumed to be **incomplete**!



Important

Ontologies are logical theories $\rightsquigarrow$
They are perfectly suited to deal with **incomplete information**!



Query answering with incomplete information

In the presence of incomplete information:

- Query answering amounts to computing **certain answers**, given the global view, the mapping and the data at the sources.
- Query answering may be **costly** (even with a single data source and with the simplest forms of mappings).

We look into query answering over ontologies, abstracting from the data sources (we consider the data therein as the extensional level of the ontology).

unibz.it

Outline

unibz.it

Query answering under different assumptions

Depending on the setting, query answering may have different meanings:

- Traditional databases $\leadsto$ **complete information**
- Ontologies (or knowledge bases) $\leadsto$ **incomplete information**

unibz.it

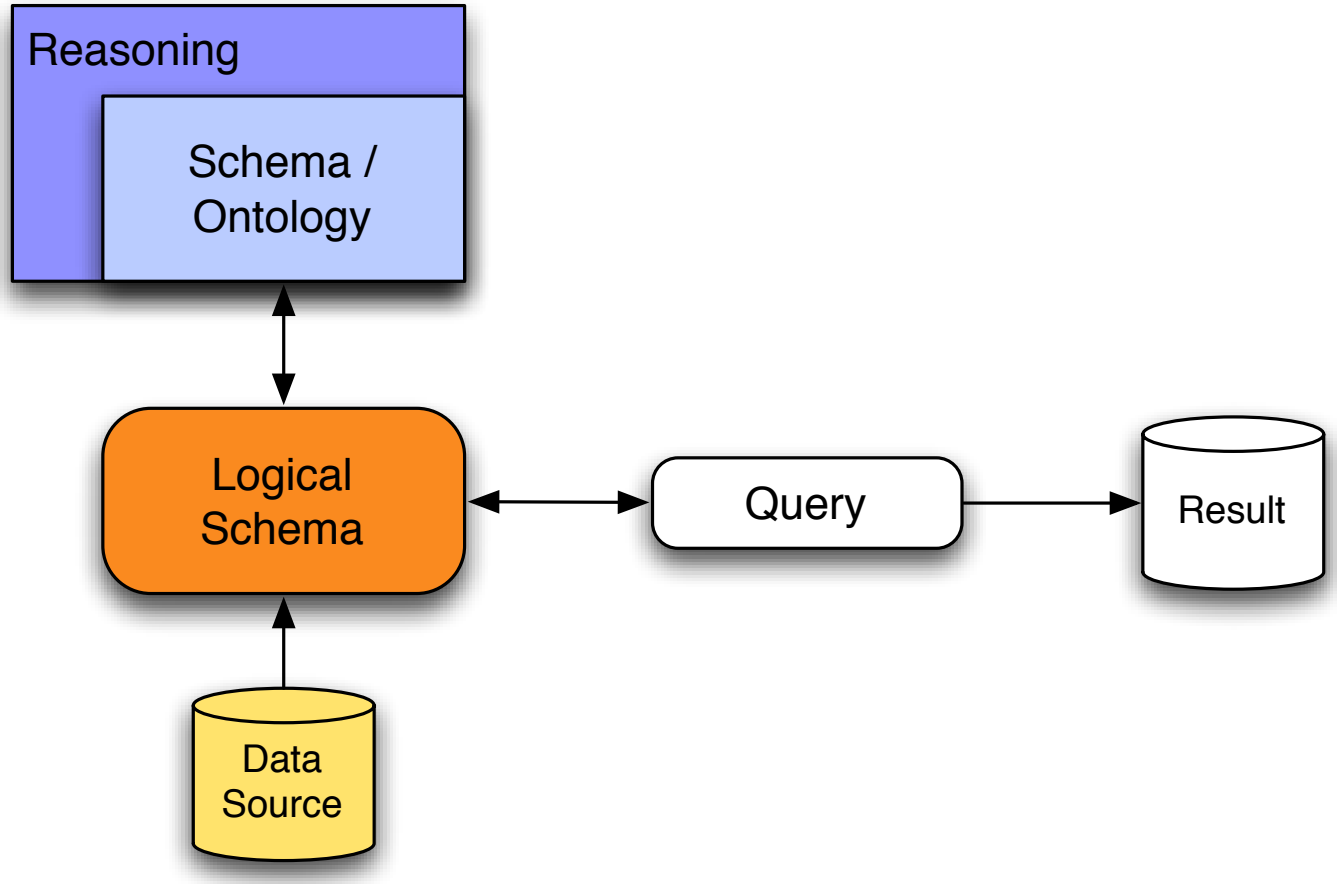
Query answering in traditional databases

- Data are completely specified (CWA), and typically large.
- Schema/intensional information used in the design phase.
- At **runtime**, the data is assumed to satisfy the schema, and therefore the **schema is not used**.
- Queries may express complex conditions on the data (cf. SQL).

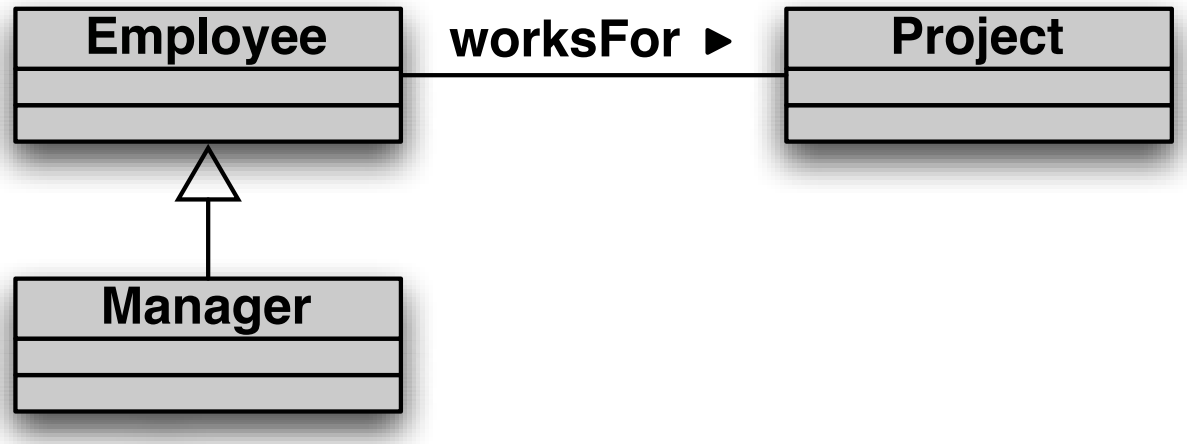
$\rightsquigarrow$ Query answering amounts to **query evaluation**, which is computationally easy.

unibz.it

Query answering in traditional databases (cont'd)



Query answering in traditional databases – Example



For each concept/relationship we have a (complete) table in the DB.

DB: Employee $= \{$ `ernest`, `oscar`, `alberto` $\}$

Manager $= \{$ `ernest`, `alberto` $\}$

Project $= \{$ `einagmc`, `folre` $\}$

worksFor $= \{$ (`ernest`,`einagmc`), (`oscar`,`folre`) $\}$

Query: $q(x) \leftarrow \exists p.\, \mathsf{Manager}(x), \mathsf{Project}(p), \mathsf{worksFor}(x, p)$

Answer: $\{$ `ernest` $\}$

unibz.it

Query answering over ontologies

- An ontology imposes constraints on the data.
- Actual data may be incomplete or inconsistent w.r.t. such constraints.
- The system has to take into account the constraints during query answering, and overcome incompleteness or inconsistency.

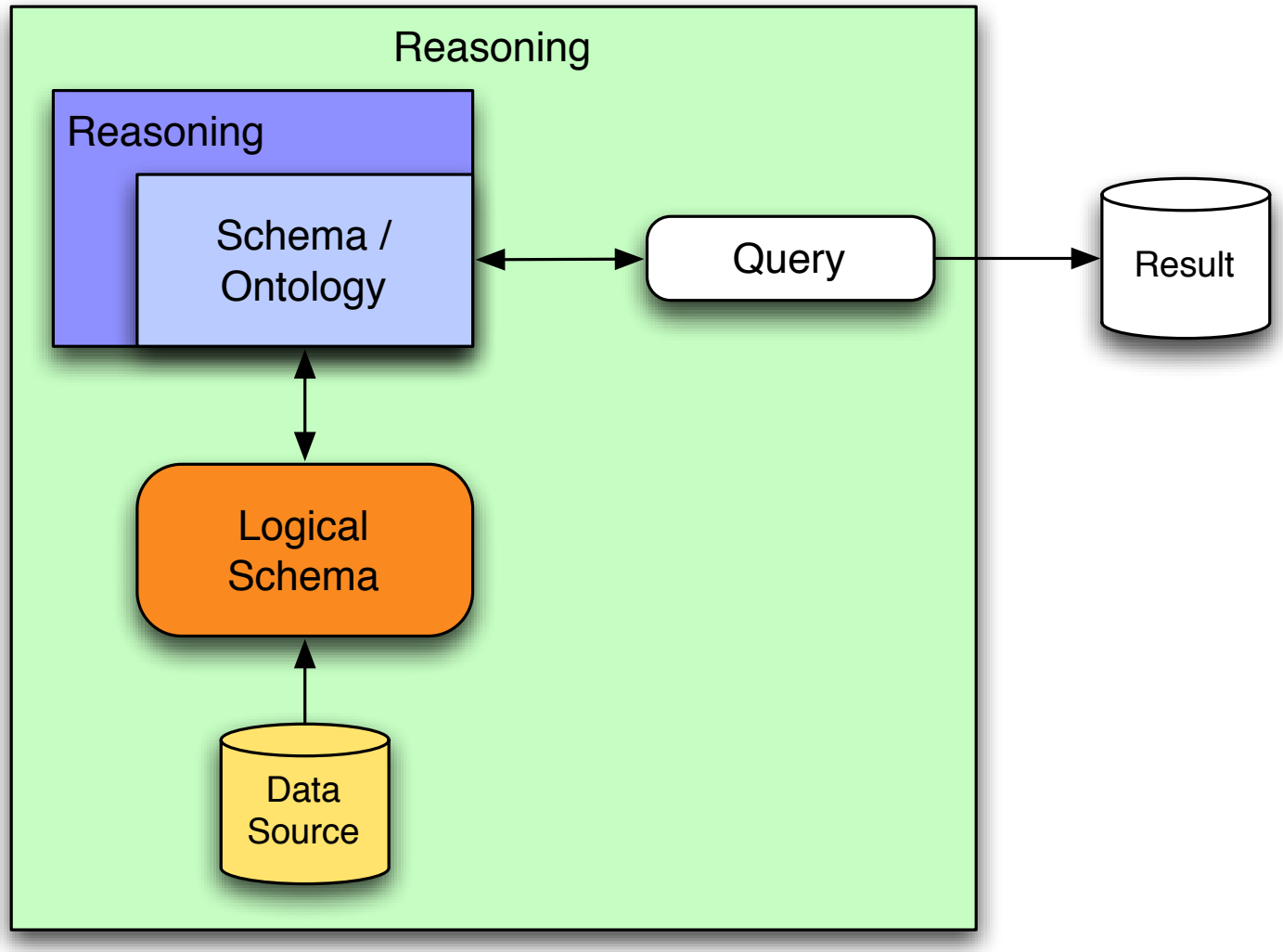
$\rightsquigarrow$ Query answering amounts to **logical inference**, which is computationally more costly.

Note:

- Size of the data is not considered critical (comparable to the size of the intensional information).
- Queries are typically simple, i.e., atomic (a class name), and query answering amounts to instance checking.

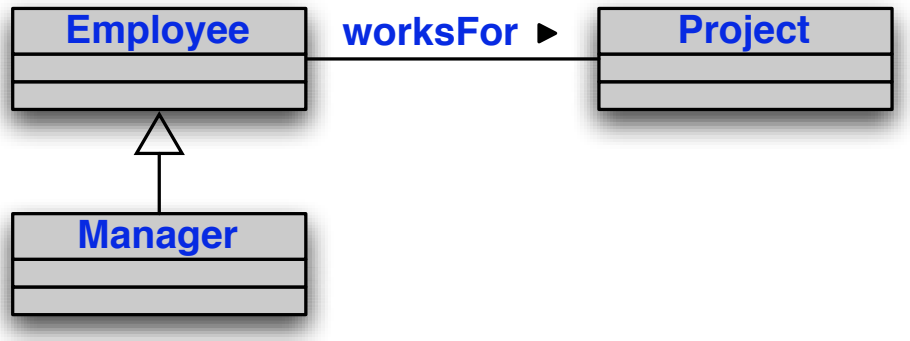
unibz.it

Query answering over ontologies (cont'd)



unibz.it

Query answering over ontologies – Example



The tables in the database may be **incompletely specified**, or even missing for some classes/properties.

DB: Manager $\supseteq$ { ernest, alberto }
Project $\supseteq$ { einagmc, folre }
worksFor $\supseteq$ { (ernest,einagmc), (oscar,folre) }

Query: $q(x) \leftarrow \text{Employee}(x)$

Answer: { ernest, alberto, oscar }

Ontology-based data access (and integration)

In ontology-based data access (and integration), we have to face the difficulties of both settings:

- The actual **data** is stored in external sources (i.e., databases), and thus its size is typically **very large**.
- The ontology introduces **incompleteness** of knowledge, and we have to do logical inference, rather than query evaluation.
- We want to take into account at **runtime** the **constraints** expressed in the ontology.
- We want to answer **complex database-like queries**.

Note: In OBDI, we have to deal with multiple information sources, i.e., we face also the problems that are typical of data integration.

unibz.it

Questions that need to be addressed

1. Which is the "right" **query language**?
2. Which is the "right" **ontology language**?
3. How can we bridge the **semantic mismatch** between the ontology and the data sources?
4. How can **tools for ontology-based data access and integration** take into account these issues?

We propose an approach based on Description Logics that provides answers to these questions.

unibz.it

Outline

unibz.it

Description Logics

- **Description Logics (DLs)** stem from early days (70') KR formalisms, such as Semantic Networks and Frame Systems, and assumed their current form in the late 80's & 90's. They strongly influenced W3C OWL (OWL-DL is a DL!).
- They are **logics** specifically designed to represent and reason on structured knowledge:
  - domain of interest represented as composed of objects structured into:
    - concepts, corresponding to classes, and denoting sets of objects;
    - roles, corresponding to (binary) relationships, and denoting binary relations on objects.
  - knowledge is predicated through so-called assertions, i.e., logical axioms.
- Technically they can be considered well-behaved (i.e., decidable) **fragments of first-order logic**.

unibz.it

DL ontology (or knowledge base)

Is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

Description Logic **TBox** $\mathcal{T}$

Represents the **intensional** *knowledge in the ontology*.

Consists of a set of universal assertions on concepts and roles:

- Inclusion assertions on concepts: $C_1 \sqsubseteq C_2$ $\quad [\forall x(C_1(x) \rightarrow C_2(x))]$
- Inclusion assertions on roles: $R_1 \sqsubseteq R_2$ $\quad [\forall x, y(R_1(x, y) \rightarrow R_2(x, y))]$
- Property assertions on (atomic) roles:
  (**functional** R) $\quad$ (**transitive** R) $\quad \cdots$

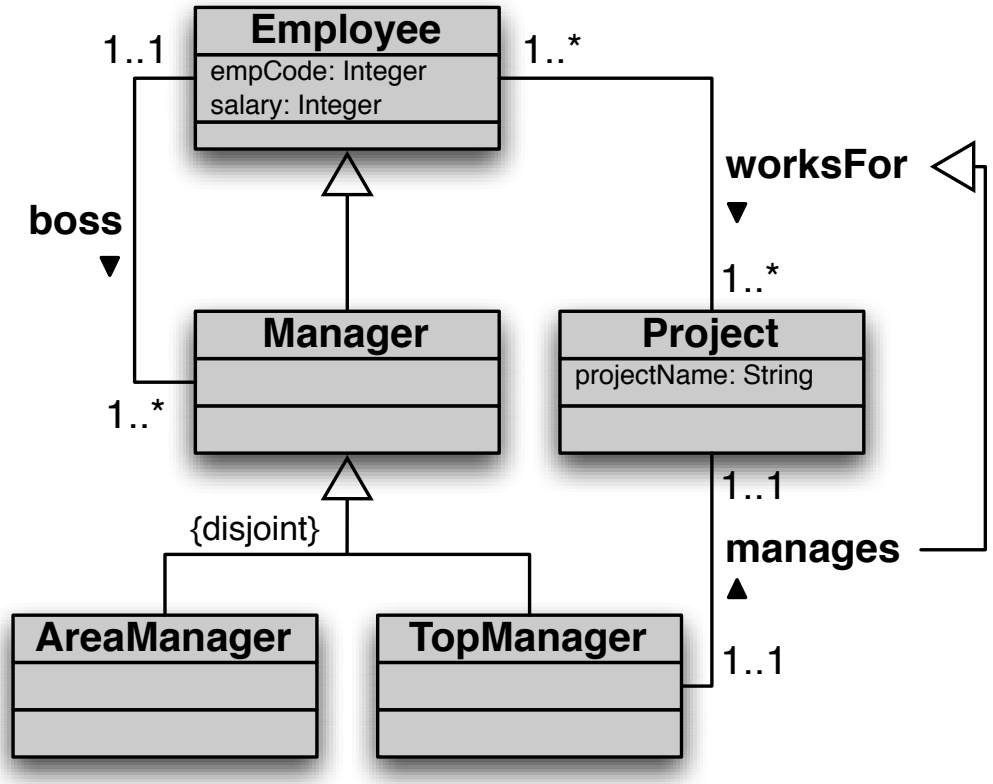
Description Logic **ABox** $\mathcal{A}$

Represents the **extensional** *knowledge in the ontology*.

Consists of a set of membership assertions on individuals:

- for concepts: $A(c)$
- for roles: $P(c_1, c_2)$ (we use c_i to denote individuals)

Example of DL ontology



TBox:

$$
\begin{aligned}
\mathit{Manager} &\sqsubseteq \mathit{Employee}\\
\mathit{AreaManager} &\sqsubseteq \mathit{Manager}\\
\mathit{TopManager} &\sqsubseteq \mathit{Manager}\\
\mathit{AreaManager} &\sqsubseteq \neg \mathit{TopManager}\\
\\
\mathit{Employee} &\sqsubseteq \exists \mathit{salary}\\
\exists \mathit{salary}^- &\sqsubseteq \texttt{xsd:int}\\
(\textbf{funct}\ \mathit{salary})&\\
\\
\exists \mathit{worksFor} &\sqsubseteq \mathit{Employee}\\
\exists \mathit{worksFor}^- &\sqsubseteq \mathit{Project}\\
\mathit{Employee} &\sqsubseteq \exists \mathit{worksFor}\\
\mathit{Project} &\sqsubseteq \exists \mathit{worksFor}^-\\
\\
\exists \mathit{manages} &\sqsubseteq \mathit{TopManager}\\
\exists \mathit{manages}^- &\sqsubseteq \mathit{Project}\\
\mathit{TopManager} &\sqsubseteq \exists \mathit{manages}\\
\mathit{Project} &\sqsubseteq \exists \mathit{manages}^-\\
\mathit{manages} &\sqsubseteq \mathit{worksFor}\\
(\textbf{funct}\ \mathit{manages})&\\
(\textbf{funct}\ \mathit{manages}^-)&\\
\ldots&
\end{aligned}
$$

ABox:

Manager(ernest)
manages(ernest,einagmc)
. . .

unibz.it

Which language to use for querying ontologies?

Two borderline cases:

1. Just classes and properties of the ontology $\leadsto$ instance checking
   - Ontology languages are tailored for capturing intensional relationships.
   - They are quite **poor as query languages**:
     Cannot refer to same object via multiple navigation paths in the ontology, i.e., allow only for a limited form of JOIN, namely chaining.

2. Full SQL (or equivalently, first-order logic)
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

Unions of conjunctive queries

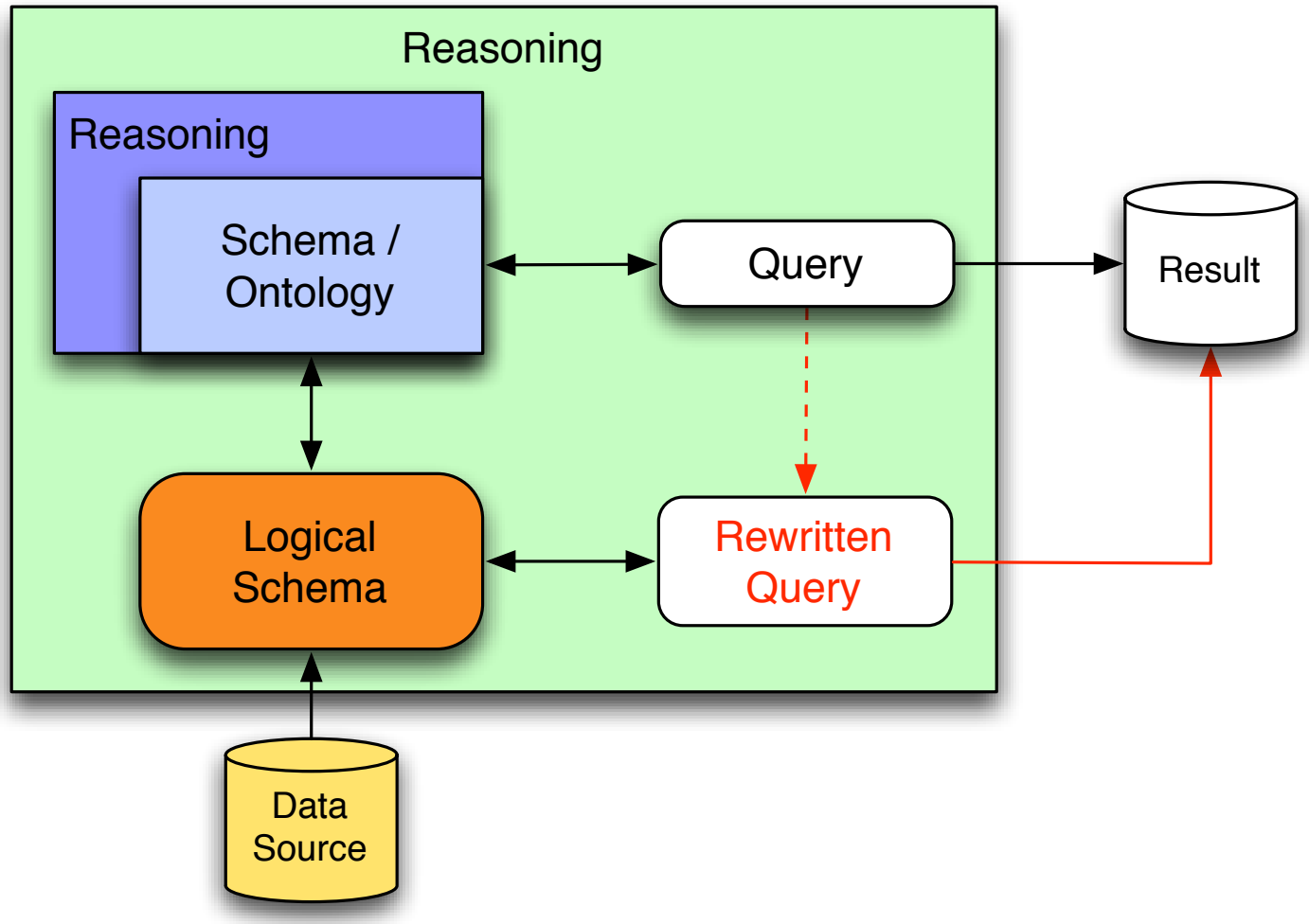
A good tradeoff is to use (unions of) **conjunctive queries** (UCQs):

- A (U)CQ is a first-order query using only conjunction, existential quantification (and disjunction).
- Hence, UCQs contain no negation, no universal quantification, and no function symbols besides constants.
- Correspond to SQL/relational algebra **(union) select-project-join (SPJ) queries** – the most frequently asked queries.
- For (U)CQs over an ontology, the predicates in atoms are concepts and roles of the ontology.

Query answering amounts to finding the answers to $q(\vec{x})$ that hold in all models of an ontology $\mathcal{O} \rightsquigarrow$ **Certain answers** $cert(q, \mathcal{O})$ to q over $\mathcal{O}$.

unibz.it

Query answering via rewriting



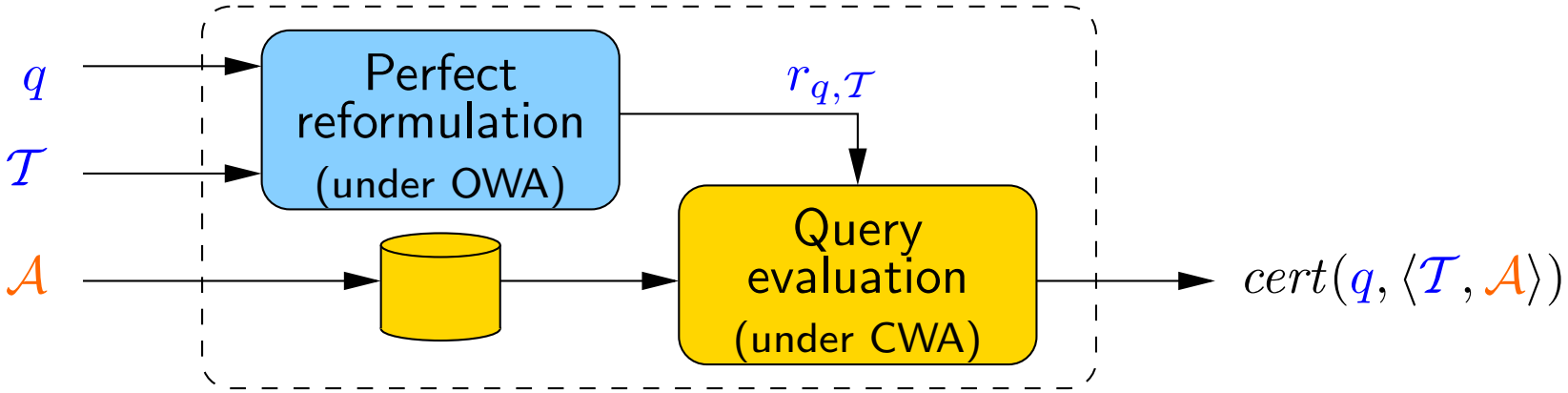
unibz.it

Inference in query answering

To be able to deal with data efficiently, we need to separate the contribution of $\mathcal{A}$ from the contribution of q and $\mathcal{T}$.

$\rightsquigarrow$ Query answering by **query rewriting**.

Query rewriting



Query answering can **always** be thought as done in two phases:

1. **Perfect rewriting**: produce from q and the TBox $\mathcal{T}$ a new query $r_{q,\mathcal{T}}$ (called the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. **Query evaluation**: evaluate $r_{q,\mathcal{T}}$ over the ABox $\mathcal{A}$ seen as a complete database (and without considering the TBox $\mathcal{T}$).
   $\rightsquigarrow$ Produces $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle)$.

Note: The "always" holds if we pose no restriction on the language in which to express the rewriting $r_{q,\mathcal{T}}$.


unibz.it

Language of the rewriting

The **expressiveness of the ontology language affects the rewriting language**, i.e., the language into which we are able to rewrite UCQs:

- When we can rewrite into FOL/SQL.
  $\rightsquigarrow$ Query evaluation can be done in SQL, i.e., via an RDBMS (*Note:* FOL is in AC^0).
- When we can rewrite into an NLogSpace-hard language.
  $\rightsquigarrow$ Query evaluation requires (at least) linear recursion.
- When we can rewrite into a PTime-hard language.
  $\rightsquigarrow$ Query evaluation requires full recursion (e.g., Datalog).
- When we can rewrite into a coNP-hard language.
  $\rightsquigarrow$ Query evaluation requires (at least) power of Disjunctive Datalog.

unibz.it

Complexity of query answering in DLs

Problem of rewriting is related to **complexity of query answering**.

Studied extensively for (unions of) CQs and various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 (2)
OWL 2 (and less)	2ExpTime-complete	coNP-hard (1)

(1) Already for a TBox with a single disjunction.
(2) This is what we need to scale with the data.

Questions

- Can we find interesting (description) logics for which query answering can be done efficiently (i.e., in AC^0)?
- If yes, can we leverage relational database technology for query answering?

The *DL-Lite* family

- A family of DLs optimized according to the tradeoff between expressive power and **complexity** of query answering, with emphasis on **data**.
- Carefully designed to have nice computational properties for answering UCQs (i.e., computing certain answers):
  - The same complexity as relational databases.
  - In fact, query answering can be delegated to a relational DB engine.
  - The DLs of the *DL-Lite* family are essentially the maximally expressive ontology languages enjoying these nice computational properties.
- We look at *DL-Lite*$_{\mathcal{A}}$, an expressive member of the *DL-Lite* family.

DL-Lite$_{\mathcal{A}}$ provides robust foundations for Ontology-Based Data Access.

unibz.it

DL-Lite$_\mathcal{A}$ ontologies

TBox assertions:

- Concept inclusion assertions: $B \sqsubseteq C$, with:

$$\begin{array}{rcl} B & \longrightarrow & A \mid \exists Q \\ C & \longrightarrow & B \mid \neg B \end{array}$$

- Role inclusion assertions: $Q \sqsubseteq R$, with:

$$\begin{array}{rcl} Q & \longrightarrow & P \mid P^- \\ R & \longrightarrow & Q \mid \neg Q \end{array}$$

- Functionality assertions: (**funct** Q)
- **Proviso:** functional roles cannot be specialized.

ABox assertions: $A(c)$, $P(c_1, c_2)$, with c_1, c_2 constants

Note: *DL-Lite*$_\mathcal{A}$ distinguishes also between roles and attributes (ignored here).

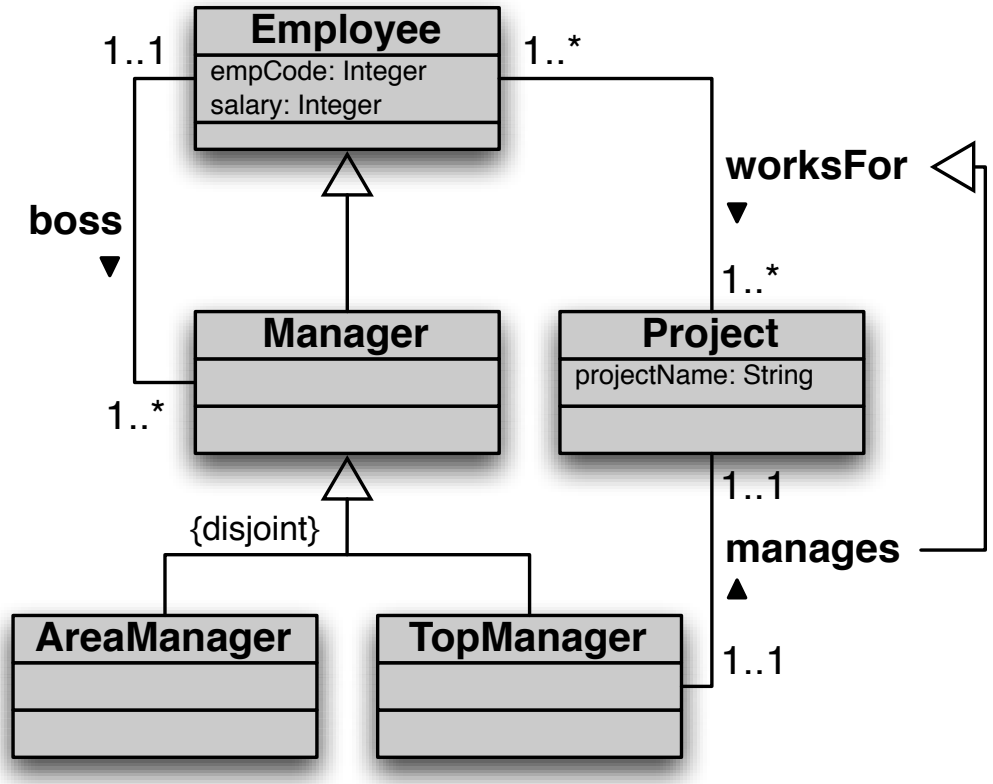
unibz.it

Capturing basic ontology constructs in *DL-Lite*$_{\mathcal{A}}$

ISA on concepts	$A_1 \sqsubseteq A_2$	$\forall x(A_1(x) \rightarrow A_2(x))$
Disj. of concepts	$A_1 \sqsubseteq \neg A_2$	$\forall x(A_1(x) \rightarrow \neg A_2(x))$
Domain of roles	$\exists P \sqsubseteq A_1$	$\forall x(\exists y(P(x,y)) \rightarrow A_1(x))$
Range of roles	$\exists P^- \sqsubseteq A_2$	$\forall x(\exists y(P(y,x)) \rightarrow A_2(x))$
Mandatory partic. *(min card = 1)*	$A_1 \sqsubseteq \exists P$	$\forall x(A_1(x) \rightarrow \exists y(P(x,y)))$
	$A_2 \sqsubseteq \exists P^-$	$\forall x(A_2(x) \rightarrow \exists y(P(y,x)))$
Role functionality *(max card = 1)*	(**funct** P)	$\forall x,y,y'(P(x,y) \wedge P(x,y') \rightarrow y = y')$
	(**funct** P^-)	$\forall x,x',y(P(x,y) \wedge P(x',y) \rightarrow x = x')$
ISA on roles	$Q_1 \sqsubseteq Q_2$	$\forall x,y(Q_1(x,y) \rightarrow Q_2(x,y))$
Disj. of roles	$Q_1 \sqsubseteq \neg Q_2$	$\forall x,y(Q_1(x,y) \rightarrow \neg Q_2(x,y))$

unibz.it

Capturing UML class diagrams/ER schemas in *DL-Lite*$_\mathcal{A}$



$$
\begin{aligned}
\text{Manager} &\sqsubseteq \text{Employee}\\
\text{AreaManager} &\sqsubseteq \text{Manager}\\
\text{TopManager} &\sqsubseteq \text{Manager}\\
\text{AreaManager} &\sqsubseteq \neg\text{TopManager}
\end{aligned}
$$

$$
\begin{aligned}
\text{Employee} &\sqsubseteq \exists \mathit{salary}\\
\exists \mathit{salary}^- &\sqsubseteq \texttt{xsd:int}\\
&(\textbf{funct}\ \mathit{salary})
\end{aligned}
$$

$$
\begin{aligned}
\exists \text{worksFor} &\sqsubseteq \text{Employee}\\
\exists \text{worksFor}^- &\sqsubseteq \text{Project}\\
\text{Employee} &\sqsubseteq \exists \text{worksFor}\\
\text{Project} &\sqsubseteq \exists \text{worksFor}^-
\end{aligned}
$$

$$
\begin{aligned}
\exists \text{manages} &\sqsubseteq \text{TopManager}\\
\exists \text{manages}^- &\sqsubseteq \text{Project}\\
\text{TopManager} &\sqsubseteq \exists \text{manages}\\
\text{Project} &\sqsubseteq \exists \text{manages}^-\\
\text{manages} &\sqsubseteq \text{worksFor}\\
&(\textbf{funct}\ \text{manages})\\
&(\textbf{funct}\ \text{manages}^-)\\
&\vdots
\end{aligned}
$$

Note: *DL-Lite*$_\mathcal{A}$ cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

Observations on *DL-Lite*$_{\mathcal{A}}$

- Captures all the basic constructs of **UML Class Diagrams** and of the **ER Model** . . .
- . . . **except covering constraints** in generalizations.
- Is at the basis of the **OWL 2 QL** profile of OWL 2.
- Extends (the DL fragment of) the ontology language **RDFS**.
- Is completely symmetric w.r.t. **direct and inverse properties**.

unibz.it

Query answering in *DL-Lite*$_{\mathcal{A}}$

Based on query reformulation: given an (U)CQ and an ontology:

1. **Compute its perfect rewriting**, which turns out to be a UCQ.
2. **Evaluate the perfect rewriting** on the ABox seen as a DB.

To compute the perfect rewriting, starting from the original (U)CQ, iteratively get a CQ to be processed and either:

- **expand** positive inclusions & **simplify** redundant atoms, or
- **unify** atoms in the CQ to obtain a more specific CQ to be further expanded.

Each result of the above steps is added to the queries to be processed.

Note: negative inclusions and functionalities play a role in ontology satisfiability, but not in query answering (i.e., we have **separability**).

unibz.it

Query answering in *DL-Lite*$_{\mathcal{A}}$ – Example

TBox:

$\mathsf{Manager} \sqsubseteq \exists \mathsf{worksFor}$ $\quad\quad$ $\forall x(\mathsf{Manager}(x) \rightarrow \exists y(\mathsf{worksFor}(x, y)))$

$\exists \mathsf{worksFor}^- \sqsubseteq \mathsf{Project}$ $\quad\quad$ $\forall x(\exists y(\mathsf{worksFor}(y, x)) \rightarrow \mathsf{Project}(x))$

Query: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$

Perfect Reformulation: $\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$

$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{worksFor}(\_, y)$

$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, \_)$

$\mathsf{q}(x) \leftarrow \mathsf{Manager}(x)$

ABox: $\mathsf{worksFor}(\mathtt{ernest}, \mathtt{einagmc})$ $\quad\quad$ $\mathsf{Manager}(\mathtt{ernest})$

$\mathsf{worksFor}(\mathtt{oscar}, \mathtt{folre})$ $\quad\quad$ $\mathsf{Manager}(\mathtt{alberto})$

Evaluating the last two queries over the ABox (seen as a DB) produces as answer $\{\mathtt{ernest}, \mathtt{oscar}, \mathtt{alberto}\}$.

unibz.it

Complexity of reasoning in *DL-Lite*$_{\mathcal{A}}$

Ontology satisfiability and all classical DL reasoning tasks are:

- Efficiently tractable in the size of the TBox (i.e., PTIME).
- Very efficiently tractable in the size of the ABox (i.e., AC^0).

In fact, reasoning can be done by constructing suitable FOL/SQL queries and evaluating them over the ABox (**FOL-rewritability**).

Query answering for CQs and UCQs is:

- PTIME in the size of the TBox.
- AC^0 in the size of the ABox.
- Exponential in the size of the **query** (NP-complete).
  Bad? ... not really, this is exactly as in relational DBs.

Can we go beyond *DL-Lite*$_{\mathcal{A}}$?

No! By adding essentially any additional constructor we lose these nice computational properties [CDGL$^+$06].

.it

Beyond *DL-Lite*$_\mathcal{A}$: results on data complexity

	Lhs	Rhs	funct.	Role incl.	Data complexity of query answering
0	*DL-Lite*$_\mathcal{A}$		$\surd$*	$\surd$*	in AC^0
1	$A \mid \exists P.A$	A	–	–	NLogSpace-hard
2	A	$A \mid \forall P.A$	–	–	NLogSpace-hard
3	A	$A \mid \exists P.A$	$\surd$	–	NLogSpace-hard
4	$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTime-hard
5	$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTime-hard
6	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	$\surd$	–	PTime-hard
7	$A \mid \exists P.A \mid \exists P^-.A$	$A \mid \exists P$	–	–	PTime-hard
8	$A \mid \exists P \mid \exists P^-$	$A \mid \exists P \mid \exists P^-$	$\surd$	$\surd$	PTime-hard
9	$A \mid \neg A$	A	–	–	coNP-**hard**
10	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-**hard**
11	$A \mid \forall P.A$	A	–	–	coNP-**hard**

Notes:

- * with the "proviso" of not specializing functional properties.
- NLogSpace and PTime hardness holds already for instance checking.
- For coNP-hardness in line 10, a TBox with a single assertion $A_L \sqsubseteq A_T \sqcup A_F$ suffices! $\rightsquigarrow$ **No** hope of including **covering constraints**.

unibz.it

Systems

- **QuOnto** [developed at Sapienza Univ. of Rome]: *DL-Lite*$_{\mathcal{A}}$ reasoning system
  - ABox on relational DB (several RDBMS supported)
  - sound and complete reformulation process implemented in Java
  - several nontrivial optimizations (based on query containment)
  - evaluation of reformulated queries delegated to RDBMS
  - scales up to ontologies containing millions of facts in the ABox (limitations in ABox size come from current relational DBMS)
- **OBDA Protégé Plugin** [developed at FUB]: Protégé 4 interface for QuOnto
  - based on OWL-DL
  - includes OWL and SPARQL translators
  - makes use of an extended OWL API to handle queries
- **OWLgres** [developed by Clark&Parsia]: implements a subset of *DL-Lite*$_{\mathcal{A}}$
  - ABox is represented in a relational DB in Postgres
  - accesses the data to optimize reformulation
  - current implementation is sound but incomplete

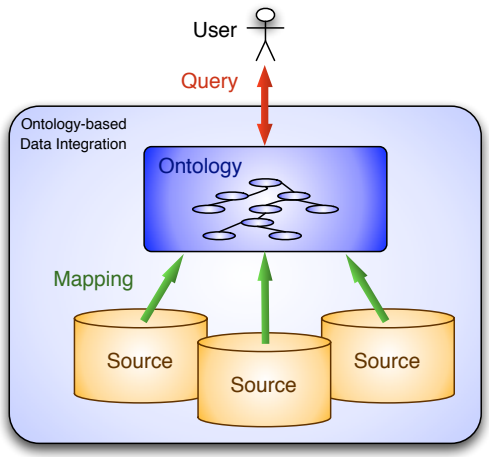
Note: *DL-Lite* has been standardized as the OWL 2 QL Profile of OWL 2.

unibz.it

Outline

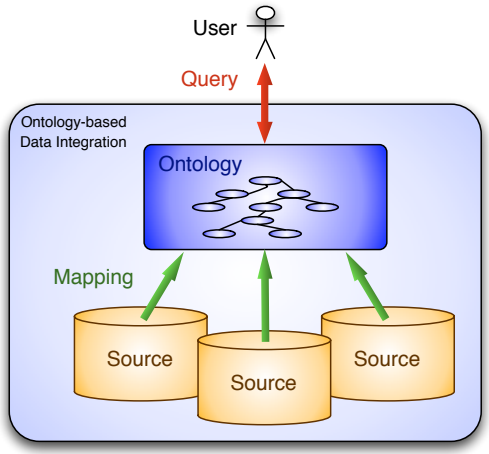
unibz.it

Ontology-based data integration: The *DL-Lite*$_\mathcal{A}$ solution



- We require the data sources to be **wrapped** and presented as relational sources. $\rightsquigarrow$ *"Standard technology"*
- We make use of a **data federation tool**, such as IBM Information Integrator, to present the yet to be (semantically) integrated sources as a single relational database. $\rightsquigarrow$ *"Standard technology"*
- We make use of the **DL-Lite**$_\mathcal{A}$ technology presented above for the conceptual view on the data, to **exploit effectiveness of query answering**. $\rightsquigarrow$ *"New technology"*

Mappings in ontology based data integration



- The (federated) source DB is **external** and **independent** from the conceptual view (the ontology).
- **Mappings** relate information in the sources to the ontology. $\rightsquigarrow$ sort of virtual ABox
  We use GAV (global-as-view) mappings: the result of an (arbitrary) SQL query on the source DB provides a (partial) extension of a concept/role.
- We exploit the distinction between **objects** and **values** in *DL-Lite*$_{\mathcal{A}}$ to deal with the notorious **impedance mismatch problem**!

Impedance mismatch problem

The impedance mismatch problem

- In **relational databases**, information is represented in forms of tuples of **values**.
- In **ontologies** (or more generally object-oriented systems or conceptual models), information is represented using both **objects** and values ...
  - ... with objects playing the main role, ...
  - ... and values a subsidiary role as fillers of object's attributes.

$\rightsquigarrow$ *How do we reconcile these views?*

Solution:

- We use **constructors to create objects** of the ontology from tuples of values in the DB.
- Formally, such constructors are simply Skolem functions!
- The constructors are used in the mapping queries.

unibz.it

Ontology with mappings to data sources

An **ontology with mappings** is characterized by a triple $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ such that:

- $\mathcal{T}$ is a TBox;
- $\mathcal{S}$ is a (federated) relational database representing the sources;
- $\mathcal{M}$ is a set of **mapping assertions**, each one of the form

$$\Phi(\vec{x}) \rightsquigarrow \Psi(\mathbf{f}(\vec{x}), \vec{x})$$

where

- $\Phi(\vec{x})$ is an arbitrary SQL query over $\mathcal{S}$, returning attributes $\vec{x}$
- $\Psi(\mathbf{f}(\vec{x}), \vec{x})$ is a conjunctive query over $\mathcal{T}$ without non-distinguished variables, whose variables (possibly used in terms $\mathbf{f}(\vec{x})$) are from $\vec{x}$.

We can assign to an ontology with mappings a **formal semantics**.
Note: $\mathcal{O}$ might be unsatisfiable.

unibz.it

DL-Lite$_{\mathcal{A}}$ query answering for data integration

Given a (U)CQ q and $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ (assumed to be satisfiable), we compute **certain answers** $cert(q, \mathcal{O})$ as follows:

1. Using $\mathcal{T}$, **reformulate** CQ q as a union $r_{q,\mathcal{T}}$ of CQs.
2. Using $\mathcal{M}$, **unfold** $r_{q,\mathcal{T}}$ to obtain a union $unfold(r_{q,\mathcal{T}})$ of CQs.
3. **Evaluate** $unfold(r_{q,\mathcal{T}})$ directly over $\mathcal{S}$ using RDBMS technology.

Correctness of this algorithm shows FOL-reducibility of query answering.

$\rightsquigarrow$ Query answering can again be done using **RDBMS technology**.

$\rightsquigarrow$ Prototype system implemented: Quonto+Integration Module

unibz.it

Computational complexity of query answering

Query answering in a *DL-Lite*$_{\mathcal{A}}$ ontology with mappings $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is:

- Efficiently tractable in the size of the TBox $\mathcal{T}$ and the **mappings** $\mathcal{M}$ (i.e., PTIME).
- Very efficiently tractable in the size of the **database** $\mathcal{S}$ (i.e., AC^0, and in fact FOL-rewritable).
- Exponential in the size of the **query** (NP-complete).

Can we move to LAV or GLAV mappings?

No, if we want to stay in AC^0[CDGL+08a].

Implemented system

- **Data Access and Integration Layer for QuOnto** [developed at FUB in collaboration with Sapienza Univ. of Rome]:
  - connects to external relational (federated) DB (several RDBMSs supported)
  - supports general mappings as presented here
  - sound and complete reformulation and unfolding process implemented in Java
  - evaluation of reformulated & unfolded queries delegated to RDBMS
- **ODBA Plugin for Protégé** [developed at FUB]: Protégé 4 interface to QuOnto for managing mappings and queries
  - supports the design of mappings
  - supports connection to external sources for querying
  - extends OWL API with mapping and source management capabilities

unibz.it

Experimentations and experiences

Several experimentations have been and are being carried out:

- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Monte dei Paschi di Siena
- Horizontal Gene Transfer data and ontology

Observations:

- Approach highly effective for bridging impedance mismatch.
- Rewriting technique effective against incompleteness in the data.
- Performance is still a critical aspect:
  - Optimizations in the generation of the rewriting are crucial.
  - Qualified existential quantification must be treated natively.
  - Management of mappings is crucial to performance.
  - Optimization of mapping unfolding by exploiting information on sources [RM10].

unibz.it

Outline

unibz.it

Conclusions

- Ontology-based data access and integration are challenging problems with great practical relevance.
- In this setting, the size of the data is the relevant parameter that must guide technological choices.
- Currently, scalability w.r.t. the size of the data can be achieved only by relying on commercial technologies for managing the data, i.e., relational DBMS systems.
- In order to tailor semantic technologies so as to provide a good compromise between expressivity and efficiency, requires a thorough understanding of the semantic and computational properties of the adopted formalisms.
- We have now gained such an understanding, that allowed us to study and develop good solutions to the OBDA problem.

unibz.it

Other work related to this talk

- Extensions of *DL-Lite* with additional constructs (in particular, identification constraints and denial constraints) [CDGL$^+$08b]
- Going beyond (unions) of conjunctive queries (weaker semantics wrt FOL) using dynamic CWA offered by epistemic operators [CDGL$^+$07a]
- Update on stand-alone ontologies [DGLPR06, DGLPR07]
- Privacy-aware access [CDGLR08]
- Restricting the attention to finite models only [Ros08]
- Meta-reasoning a la RDFS [DGLR08]

Ongoing and future work:

- Provenance and explanation [BCRM08]
- Write-also access: updating the data sources through an ontology
- LAV, and the new *DL-Lite* family: AC^0 query answering after NLogSpace/PTime preprocessing of data.

unibz.it

Acknowledgements

People involved in this work:

- Giuseppe De Giacomo
- Domenico Lembo
- Maurizio Lenzerini
- Marco Ruzzi
- Antonella Poggi
- Mariano Rodriguez Muro
- Riccardo Rosati
- several master and PhD students

unibz.it

References I

[BCDG05] D. Berardi, D. Calvanese, and G. De Giacomo.
Reasoning on UML class diagrams.
Artificial Intelligence, 168(1–2):70–118, 2005.

[BCRM08] A. Borgida, D. Calvanese, and M. Rodriguez-Muro.
Explanation in the *DL-Lite* family of description logics.
In *Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1440–1457. Springer, 2008.

[CDGL$^+$04] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere.
DL-Lite: Practical reasoning for rich DLs.
In *Proc. of the 17th Int. Workshop on Description Logic (DL 2004)*, volume 104 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2004.

unibz.it

References II

[CDGL+05a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Tailoring OWL for data intensive ontologies.
In *Proc. of the 1st Int. Workshop on OWL: Experiences and Directions (OWLED 2005)*, volume 188 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2005.

[CDGL+05b] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
DL-Lite: Tractable description logics for ontologies.
In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[CDGL+06] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Data complexity of query answering in description logics.
In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

unibz.it

References III

[CDGL+07a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
EQL-Lite: Effective first-order query processing in description logics.
In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, pages 274–279, 2007.

[CDGL+07b] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family.
J. of Automated Reasoning, 39(3):385–429, 2007.

[CDGL+08a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, and M. Ruzzi.
Data integration through *DL-Lite*$_{\mathcal{A}}$ ontologies.
In K.-D. Schewe and B. Thalheim, editors, *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, volume 4925 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 2008.

unibz.it

References IV

[CDGL+08b] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Path-based identification constraints in description logics.
In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 231–241, 2008.

[CDGLR07] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati.
Actions and programs over description logic ontologies.
In *Proc. of the 20th Int. Workshop on Description Logic (DL 2007)*, volume 250 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 29–40, 2007.

[CDGLR08] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati.
View-based query answering over description logic ontologies.
In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 242–251, 2008.

unibz.it

References V

[DGLPR06] G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati.
On the update of description logic ontologies at the instance level.
In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, pages 1271–1276, 2006.

[DGLPR07] G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati.
On the approximation of instance level update and erasure in description logics.
In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007)*, pages 403–408, 2007.

[DGLR08] G. De Giacomo, M. Lenzerini, and R. Rosati.
Towards higher-order *DL-Lite*.
In *Proc. of the 21st Int. Workshop on Description Logic (DL 2008)*, volume 353 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2008.

unibz.it

References VI

[PLC+08] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati.

Linking data to ontologies.

J. on Data Semantics, X:133–173, 2008.

[RM10] M. Rodríguez-Muro.

Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics.

PhD thesis, KRDB Research Centre, Free University of Bozen-Bolzano, 2010.

[Ros08] R. Rosati.

Finite model reasoning in *DL-Lite*.

In *Proc. of the 5th European Semantic Web Conf. (ESWC 2008)*, 2008.

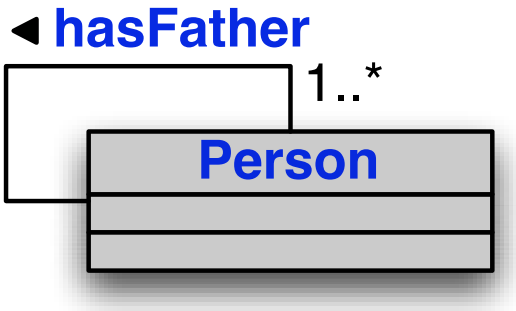
[TON08] Thinking ontologies (TONES).

EU Future and Emerging Technologies (FET) Project, 2005-2008.

`http://www.tonesproject.org/`.

unibz.it

Query answering over ontologies – Example 2



Each person has a father, who is a person.

DB: Person $\supseteq$ { jim, tom, mike }
hasFather $\supseteq$ { (jim,tom), (tom,mike) }

Queries: $q_1(x, y) \leftarrow \mathsf{hasFather}(x, y)$

$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x, y)$

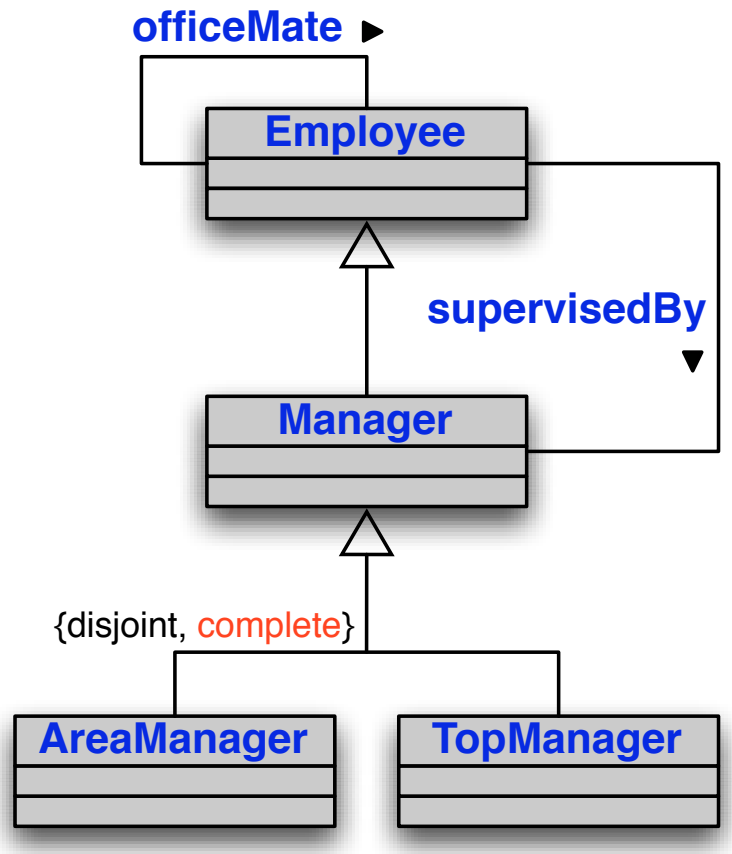
$q_3(x) \leftarrow \exists y_1, y_2, y_3.\, \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, y_3)$

$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, y_3)$

Answers: to q_1: { (jim,tom), (tom,mike) }
to q_2: { jim, tom, mike }
to q_3: { jim, tom, mike }
to q_4: { }

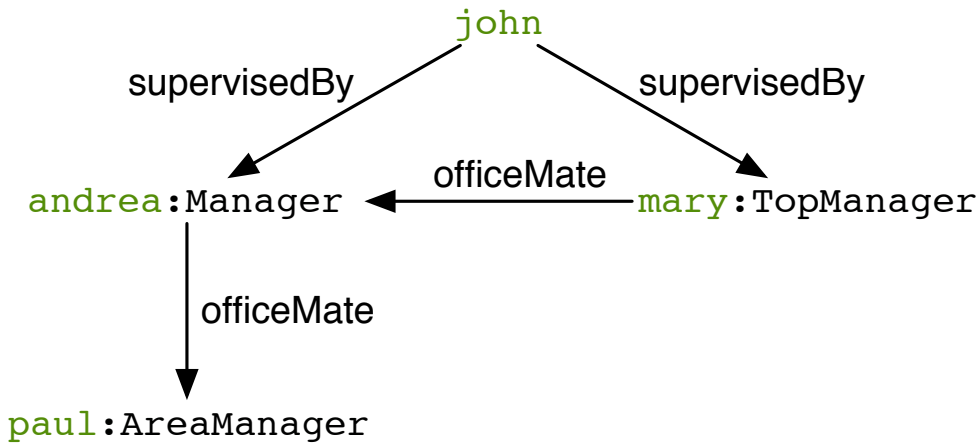
unibz.it

QA over ontologies – Andrea's Example *



$$\text{Manager} \equiv \text{AreaManager} \sqcup \text{TopManager}$$

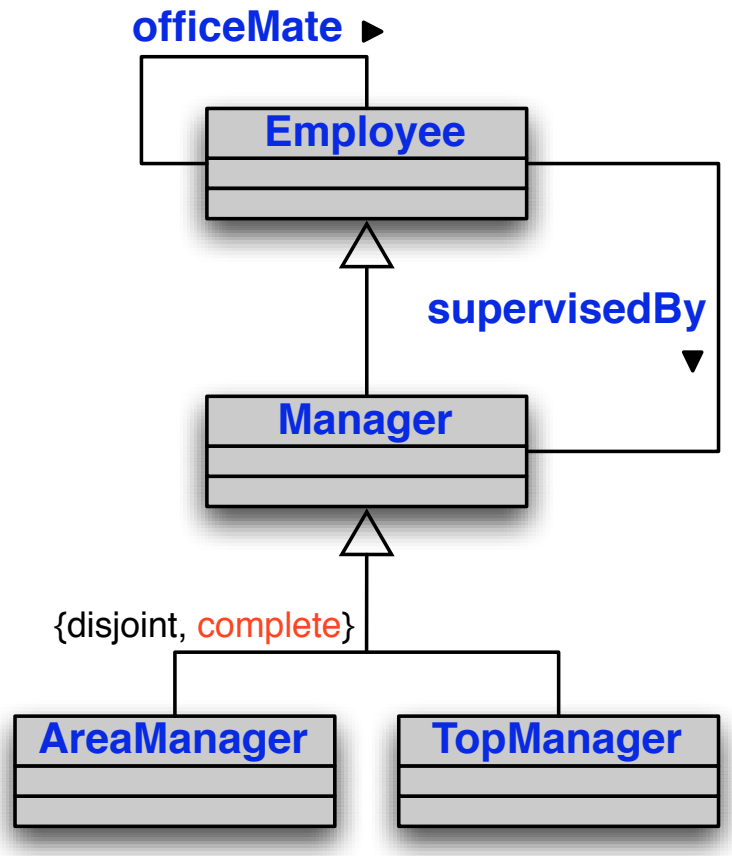
$$\begin{aligned}
\text{Employee} &\supseteq \{\ \texttt{andrea}, \texttt{paul}, \texttt{mary}, \texttt{john}\ \} \\
\text{Manager} &\supseteq \{\ \texttt{andrea}, \texttt{paul}, \texttt{mary}\ \} \\
\text{AreaManager} &\supseteq \{\ \texttt{paul}\ \} \\
\text{TopManager} &\supseteq \{\ \texttt{mary}\ \} \\
\text{supervisedBy} &\supseteq \{\ (\texttt{john},\texttt{andrea}), (\texttt{john},\texttt{mary})\ \} \\
\text{officeMate} &\supseteq \{\ (\texttt{mary},\texttt{andrea}), (\texttt{andrea},\texttt{paul})\ \}
\end{aligned}$$

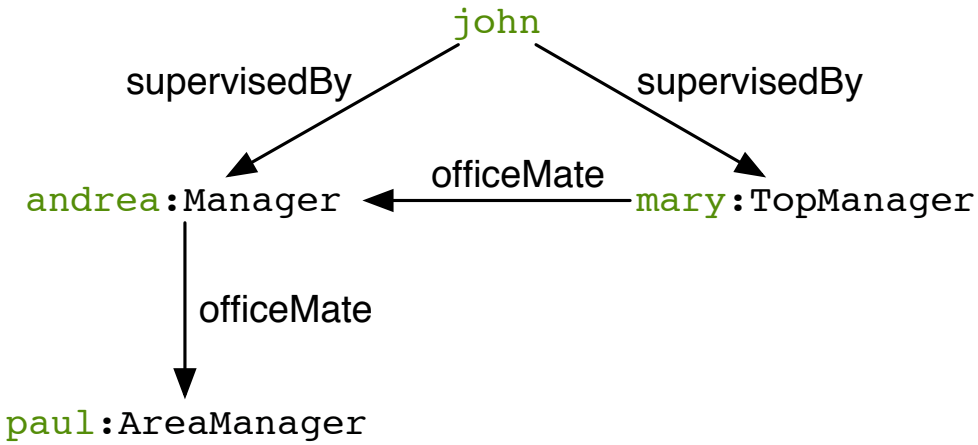


(*) By Andrea Schaerf [PhD Thesis 1994]

unibz.it

QA over ontologies – Andrea's Example (cont'd)





$$q() \leftarrow \exists x, y.\ \text{supervisedBy}(\texttt{john}, x),\ \text{TopManager}(x),\ \text{officeMate}(x, y),\ \text{AreaManager}(y)$$

Answer: **yes!**

To determine this answer, we need to resort to **reasoning by cases**.

Semantics of the *DL-Lite*$_{\mathcal{A}}$ assertions

Assertion	Syntax	Example	Semantics
class incl.	$B \sqsubseteq C$	Father $\sqsubseteq \exists$child	$B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
o-prop. incl.	$Q \sqsubseteq R$	father $\sqsubseteq$ anc	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
v.dom. incl.	$E \sqsubseteq F$	ρ(age) $\sqsubseteq$ `xsd:int`	$E^{\mathcal{I}} \subseteq F^{\mathcal{I}}$
d-prop. incl.	$U \sqsubseteq V$	offPhone $\sqsubseteq$ phone	$U^{\mathcal{I}} \subseteq V^{\mathcal{I}}$
o-prop. funct.	(**funct** Q)	(**funct** father)	$\forall o, o, o''.(o, o') \in Q^{\mathcal{I}} \wedge (o, o'') \in Q^{\mathcal{I}} \rightarrow o' = o''$
d-prop. funct.	(**funct** U)	(**funct** ssn)	$\forall o, v, v'.(o, v) \in U^{\mathcal{I}} \wedge (o, v') \in U^{\mathcal{I}} \rightarrow v = v'$
mem. asser.	$A(c)$	Father(bob)	$c^{\mathcal{I}} \in A^{\mathcal{I}}$
mem. asser.	$P(c_1, c_2)$	child(bob, ann)	$(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in P^{\mathcal{I}}$
mem. asser.	$U(c, d)$	phone(bob, '2345')	$(c^{\mathcal{I}}, \mathit{val}(d)) \in U^{\mathcal{I}}$

Note: We make the **unique-name assumption**, i.e., different constants denote different objects.

unibz.it