Optimizing Ontology-Based Data Access

Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy
Currently on sabbatical leave at Technical University Vienna, Austria
with support from Wolfgang Pauli Institute

Technical University Vienna – Faculty of Informatics
12 March 2013
Data management in information systems

Pre-DBMS architecture:

![Pre-DBMS Architecture Diagram]

Ideal architecture based on a DBMS:

![Ideal Architecture Diagram]
Data management today

In many cases, we are back at the pre-DBMS situation:

Data is:

- heterogeneous
- distributed
- redundant or even duplicated
- incoherent
Example 1: Statoil Exploration

Experts in geology and geophysics develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby geographical locations.

Facts:
- 1,000 TB of relational data
- using diverse schemata
- spread over 2,000 tables, over multiple individual data bases

Data Access for Exploration:
- 900 experts in Statoil Exploration.
- up to 4 days for new data access queries, requiring assistance from IT-experts.
- 30–70% of time spent on data gathering.
Example 2: Siemens Energy Services

Runs service centers for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components. When informed about potential problems, diagnosis engineers access a variety of raw and processed data.

Facts:

- several TB of time-stamped sensor data
- several GB of event data ("alarm triggered at time T")
- data grows at 30GB per day (sensor data rate 1Hz–1kHz)

Service Requests:

- over 50 service centers worldwide
- 1,000 service requests per center per year
- 80% of time per request used on data gathering
Proposed solution: Ontology-based Data Access

Manage data by adopting principles and techniques studied in Knowledge Representation in Artificial Intelligence.

- Based on **formalisms grounded in logic**, with well understood semantics and computational properties.
- Provide a conceptual, high level representation of the domain of interest in terms of an **ontology**.
- **Do not migrate the data** but leave it in the sources.
- **Map** the ontology to the data sources.
- Specify all information requests to the data in terms of the ontology.
- Use the inference services of the OBDA system to translate the requests into queries to the data sources.
Architecture is based on three main components:

- **Ontology**: provides a unified, conceptual view of the managed information.
- **Data source(s)**: are external and independent (possibly multiple and heterogeneous).
- **Mappings**: semantically link data at the sources with the ontology.
Ontology-based data access: Formalization

An **ontology-based access system** is a triple $\mathcal{O} = \langle T, S, M \rangle$, where:

- $T$ is the intensional level of an ontology. We consider ontologies formalized in description logics (DLs), hence $T$ is a DL TBox.
- $S$ is a (federated) relational database representing the sources;
- $M$ is a set of mapping assertions, each one of the form $\Phi(\vec{x}) \sim \Psi(\vec{x})$

where
- $\Phi(\vec{x})$ is a FOL query over $S$, returning tuples of values for $\vec{x}$
- $\Psi(\vec{x})$ is a FOL query over $T$ whose free variables are from $\vec{x}$.

**Semantics of OBDA system:**

We can assign to $\mathcal{O}$ a **first-order logic semantics**, based on interpretations of the TBox $T$, taking into account data sources $S$ and mapping $M$.

**Note:** the semantics of mappings is captured through material implication, i.e., data sources are considered **sound**, but **not necessarily complete**.
Challenges in OBDA

- How to instantiate the abstract framework?
- How to execute queries over the ontology by accessing data in the sources?
- How to deploy such systems using state-of-the-art technology?
- How to optimize the performance of the system?
- How to assess the quality of the constructed system?
- How to provide automated support for key tasks during design and deployment?
  - constructing the ontology;
  - constructing the mappings;
  - formulating queries;
  - characterizing the evolution of the system components;
  - verifying properties over the evolving system.
Instantiating the framework

1. Which is the “right” ontology language?

2. Which is the “right” query language?

3. Which is the “right” mapping language?

The choices that we make have to take into account the tradeoff between expressive power and efficiency of inference/query answering.

We are in a setting where we want to access large amounts of data, so efficiency w.r.t. the data plays an important role.
Ontologies vs. conceptual models

We leverage on an extensive amount of work on the tight relationship between conceptual modeling formalisms and ontology languages [Lenzerini and Nobili, 1990; Bergamaschi and Sartori, 1992; Borgida, 1995; C. et al., 1999; Borgida and Brachman, 2003; Berardi et al., 2005; Queralt et al., 2012].

```
Manager ⊑ Researcher
Princlnv ⊑ Manager
Coordinator ⊑ Manager
Princlnv ⊑ ¬Coordinator
Researcher ⊑ ∃salary
∃salary ⊑ xsd:int (funct salary)
∃manages ⊑ Coordinator
∃manages ⊑ Project
Coordinator ⊑ ∃manages
Project ⊑ ∃manages
manages ⊑ worksFor (funct manages) (funct manages¬) ...
```
Outline

1. Ontology-based data access framework
2. Mapping the data to the ontology
3. Query answering in OBDA
4. Ontology languages for OBDA
5. Optimizing OBDA
6. Conclusions
Use of mappings

In an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, S \rangle$, the mapping $\mathcal{M}$ is a crucial component:

- $\mathcal{M}$ encodes how the data in the external source(s) $S$ should be used to populate the elements of $\mathcal{T}$.
- We should talk about OBDA only when we are in the presence of a system that includes external sources and mappings.

Virtual data layer

The data sources $S$ and the mapping $\mathcal{M}$ define a virtual data layer $\mathcal{V} = \mathcal{M}(S)$ (i.e., a virtual ABox, in DL terminology).

- Queries are answered w.r.t. $\mathcal{T}$ and $\mathcal{V}$.
- We do not really materialize the data of $\mathcal{V}$ (that’s why it is called virtual).
- Instead, the intensional information in $\mathcal{T}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{T}$ into queries formulated over $S$. 
The impedance mismatch problem

We need to address the **impedance mismatch** problem

- In *relational databases*, information is represented as tuples of *values*.
- In *ontologies*, information is represented using both *objects* and values . . .
  - . . . with objects playing the main role, . . .
  - . . . and values palying a subsidiary role as fillers of object attributes.

Proposed solution:

- Use **constructors to create objects** of the ontology from tuples of values in the DB.
- The constructors are modeled through Skolem functions in the query in the rhs of the mapping:

  \[ \Phi(\overrightarrow{x}) \sim \Psi(\overrightarrow{f}, \overrightarrow{x}) \]

- Techniques from partial evaluation of logic programs are adapted for unfolding queries over \( \mathcal{T} \), by using \( \mathcal{M} \), into queries over \( S \).
Impedance mismatch – Example

Actual data is stored in a DB:
– A researcher is identified by her SSN.
– A project is identified by its name.

\[ D_1[SSN: \text{String}, \text{PrName}: \text{String}] \]
Researchers and projects they work for

\[ D_2[\text{Code}: \text{String}, \text{Salary}: \text{Int}] \]
Researchers’ code with salary

\[ D_3[\text{Code}: \text{String}, \text{SSN}: \text{String}] \]
Researchers’ Code with SSN

Intuitively:
• A researcher should be created from her SSN: \text{person}(SSN)
• A project should be created from its name: \text{proj}(PrName)
Mapping assertions – Example

D₁[SSN: String, PrName: String]
Researchers and Projects they work for

D₂[Code: String, Salary: Int]
Researchers’ code with salary

D₃[Code: String, SSN: String]
Researchers’ code with SSN

\[ m_1: \text{SELECT SSN, PrName } \quad \sim \quad \text{Researcher(person(SSN)), Project(proj(PrName)), projectName(proj(PrName), PrName), worksFor(person(SSN), proj(PrName))} \]

\[ m_2: \text{SELECT SSN, Salary } \quad \sim \quad \text{Researcher(person(SSN)), salary(person(SSN), Salary)} \]
Outline

1. Ontology-based data access framework
2. Mapping the data to the ontology
3. Query answering in OBDA
4. Ontology languages for OBDA
5. Optimizing OBDA
6. Conclusions
Incomplete information

We are in a setting of **incomplete information!!!**

Incompleteness introduced:

- by data source(s), in general assumed to be incomplete;
- by domain constraints encoded in the ontology.

**Plus:** Ontologies are logical theories, and hence perfectly suited to deal with incomplete information!

**Minus:** Query answering amounts to **logical inference**, and hence is significantly more challenging.
Incomplete information – Example

Assume for simplicity that each table in the underlying database is mapped directly to an ontology concept/relationship.

But the database tables may be *incompletely specified*, or even missing for some concepts/relationships.

**DB:**
- **Coordinator** ⊇ \{ serge, marie \}
- **Project** ⊇ \{ webdam, diadem \}
- **worksFor** ⊇ \{ (serge,webdam), (georg,diadem) \}

**Query:**  \( q(x) \leftarrow \text{Researcher}(x) \)

**Answer:** \{ serge, marie, georg \}
QA over ontologies – Andrea’s Example

Manager \equiv \PrincInv \sqcup \text{Coordinator}

Researcher \supseteq \{ \text{andrea, paul, mary, john} \}
Manager \supseteq \{ \text{andrea, paul, mary} \}
\PrincInv \supseteq \{ \text{paul} \}
Coordinator \supseteq \{ \text{mary} \}

\text{supervisedBy} \supseteq \{ (\text{john, andrea}), (\text{john, mary}) \}
\text{officeMate} \supseteq \{ (\text{mary, andrea}), (\text{andrea, paul}) \}

\text{john: Manager} \quad \text{officeMate} \quad \text{mary: Coordinator}
\text{supervisedBy}
\text{supervisedBy}
\text{supervisedBy}

\text{andrea: Manager} \quad \text{officeMate} \quad \text{paul: PrincInv}

(*) By Andrea Schaerf [PhD Thesis 1994]
QA over ontologies – Andrea’s Example (cont’d)

To obtain this answer, we need to reason by cases.
Query answering

Certain answers

Query answering amounts to finding the certain answers \( cert(q, \mathcal{O}) \) to a query \( q(\vec{x}) \), i.e., those answers that hold in all models of the OBDA system \( \mathcal{O} \).

Two borderline cases for the language to use for querying ontologies:

1. **Use the ontology language as query language.**
   - Ontology languages are tailored for capturing intensional relationships.
   - They are quite poor as query languages.

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

Conjunctive queries

A good tradeoff is to use conjunctive queries (CQs) or unions of CQs (UCQs), corresponding to SQL/relational algebra (union) select-project-join queries.
Complexity of conjunctive query answering in DLs

Studied extensively for various ontology languages:

<table>
<thead>
<tr>
<th></th>
<th>Combined complexity</th>
<th>Data complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain databases</td>
<td>NP-complete</td>
<td>in AC$^0$</td>
</tr>
<tr>
<td>Expressive DLs</td>
<td>$\geq 2\text{ExpTime}$</td>
<td>coNP-hard</td>
</tr>
</tbody>
</table>

(1) This is what we need to scale with the data.

(2) Hardness by [Lutz, 2008; Eiter et al., 2009].
Tight upper bounds obtained for a variety of expressive DLs [C. et al., 1998; Levy and Rousset, 1998; C. et al., 2007c; C. et al., 2008c; Glimm et al., 2008b; Glimm et al., 2008a; Lutz, 2008; Eiter et al., 2008].

(3) Already for an ontology with a single axiom involving disjunction. However, the complexity does not increase even for very expressive DLs [Ortiz et al., 2006; Ortiz et al., 2008; Glimm et al., 2008a].
Challenges for query answering in the OBDA setting

**Challenges**

- Can we find interesting ontology languages for which query answering in OBDA can be done efficiently (i.e., in AC$^0$)?

- If yes, can we delegate query answering in OBDA to a relational engine?

- If yes, can we obtain acceptable performance in practical scenarios involving large ontologies and large amounts of data?
Logical inference for query answering

Logical inference

\[ q \rightarrow \]
\[ T \rightarrow \]
\[ M(S) \rightarrow \]
\[ \text{Logical inference} \rightarrow \]
\[ \text{cert}(q, \langle T, M, S \rangle) \]

To be able to deal with data efficiently, we need to separate the contribution of the data \( S \) (accessed via the mapping \( M \)) from the contribution of \( q \) and \( O \).

\[ \sim \rightarrow \text{Query answering by \textbf{query rewriting}.} \]
Query answering by rewriting

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

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

Note: The “always” holds if we pose no restriction on the language in which to express the rewriting $r_{q,T}$. 
**$\mathcal{L}_Q$-rewritability**

Let:

- $\mathcal{L}_Q$ be a query language (i.e., a class of queries), and
- $\mathcal{L}_T$ be an ontology TBox language.

**Def.: $\mathcal{L}_Q$-rewritability of conjunctive query answering**

Conjunctive query answering is $\mathcal{L}_Q$-rewritable in $\mathcal{L}_T$, if for every TBox $\mathcal{T}$ of $\mathcal{L}_T$ and for every conjunctive query $q$, the perfect rewriting $r_{q,\mathcal{T}}$ of $q$ w.r.t. $\mathcal{T}$ can be expressed in $\mathcal{L}_Q$.

**Note:** When the only relevant measure is the size of the data $\mathcal{M}(S)$, then

$$\text{complexity of computing } \text{cert}(q, \langle \mathcal{T}, \mathcal{M}, S \rangle) = \text{complexity of evaluating } r_{q,\mathcal{T}} \text{ over } \mathcal{M}(S)$$

Hence, $\mathcal{L}_Q$-rewritability is tightly related to the data complexity of evaluating queries expressed in the language $\mathcal{L}_Q$. 
The **expressiveness of the ontology language affects the rewriting language**, i.e., the language into which we are able to rewrite (U)CQs:

- When we can rewrite into **FOL/SQL**.
  - Query evaluation can be done in SQL, i.e., via an **RDBMS**. *(Note: FOL is in AC^0).*

- When we can rewrite into **UCQs**.
  - Query evaluation can be “optimized” via an **RDBMS**.

- When we can rewrite into **non-recursive Datalog**.
  - Query evaluation can be done via an **RDBMS**, but using views.

- When we need an **NLogSpace-hard** language to express the rewriting.
  - Query evaluation requires (at least) **linear recursion**.

- When we need a **PTime-hard** language to express the rewriting.
  - Query evaluation requires full recursion (e.g., **Datalog**).

- When we need a **coNP-hard** language to express the rewriting.
  - Query evaluation requires (at least) the power of **Disjunctive Datalog**.
Outline

1. Ontology-based data access framework
2. Mapping the data to the ontology
3. Query answering in OBDA
4. Ontology languages for OBDA
5. Optimizing OBDA
6. Conclusions
Description Logics

- **Description Logics (DLs)** stem from early days (70’s) KR formalisms, and assumed their current form in the late 80’s & 90’s.

- Are **logics** specifically designed to represent and reason on structured knowledge.

- Technically they can be considered as well-behaved (i.e., decidable) fragments of first-order logic.

- Semantics given in terms of first-order interpretations.

- Come in hundreds of variations, with different semantic and computational properties.

- Strongly influenced the W3C standard Web Ontology Language OWL.
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.
  - The same complexity as relational databases.
  - In fact, **query answering is FOL-rewritable** and hence can be delegated to a relational DB engine.
  - The DLs of the **DL-Lite** family are essentially the maximally expressive DLs enjoying these nice computational properties.

- Nevertheless they have the “right” expressive power: capture the essential features of conceptual modeling formalisms.

**DL-Lite** provides robust foundations for Ontology-Based Data Access.

**Note:**

- The **DL-Lite** family is at the basis of the **OWL 2 QL profile** of the W3C standard Web Ontology Language OWL.
- More recently, the **DL-Lite** family has been extended towards $n$-ary relations and with additional features (see, e.g., [Calì et al., 2009; Baget et al., 2011; Gottlob and Schwentick, 2012]).
**DL-Lite ontologies (essential features)**

**Concept and role language:**
- Roles $R$: either atomic: $P$
  or an inverse role: $P^-$
- Concepts $C$: either atomic: $A$
  or the projection of a role on one component: $\exists P$, $\exists P^-$

**TBox assertions:**
- Role inclusion: $R_1 \sqsubseteq R_2$
- Role disjointness: $R_1 \sqsubseteq \neg R_2$
- Role functionality: $(\text{funct } R)$
- Concept inclusion: $C_1 \sqsubseteq C_2$
- Concept disjointness: $C_1 \sqsubseteq \neg C_2$

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

*Note:* DL-Lite distinguishes also between abstract objects and data values (ignored here).
## Semantics of DL-Lite

**DL-Lite** (as all DLs) is equipped with a set-theoretic semantics.

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
<th>Example</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic role</td>
<td>$P$</td>
<td>manages</td>
<td>$P^I \subseteq \Delta^I \times \Delta^I$</td>
</tr>
<tr>
<td>inverse role</td>
<td>$P^-$</td>
<td>manages$^-$</td>
<td>${(o, o') \mid (o', o) \in P^I}$</td>
</tr>
<tr>
<td>role negation</td>
<td>$\neg R$</td>
<td>$\neg$worksFor</td>
<td>$(\Delta^I \times \Delta^I) \setminus R^I$</td>
</tr>
<tr>
<td>atomic concept</td>
<td>$A$</td>
<td>Researcher</td>
<td>$A^I \subseteq \Delta^I$</td>
</tr>
<tr>
<td>existential restriction</td>
<td>$\exists R$</td>
<td>$\exists$worksFor$^-$</td>
<td>${o \mid \exists o'. (o, o') \in R^I}$</td>
</tr>
<tr>
<td>concept negation</td>
<td>$\neg C$</td>
<td>$\neg \exists$worksFor</td>
<td>$\Delta^I \setminus C^I$</td>
</tr>
<tr>
<td>concept incl.</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Project $\sqsubseteq \exists$worksFor$^-$</td>
<td>$C_1^I \subseteq C_2^I$</td>
</tr>
<tr>
<td>role incl.</td>
<td>$R_1 \sqsubseteq R_2$</td>
<td>manages $\sqsubseteq$ worksFor</td>
<td>$R_1^I \subseteq R_2^I$</td>
</tr>
<tr>
<td>role funct.</td>
<td>$\text{(funct } R\text{)}$</td>
<td>$\text{(funct } manages\text{)}$</td>
<td>$\forall o, o, o''. (o, o') \in R^I \land (o, o'') \in R^I \rightarrow o' = o''$</td>
</tr>
<tr>
<td>mem. asser.</td>
<td>$A(c)$</td>
<td>Manager(ann)</td>
<td>$c^I \in A^I$</td>
</tr>
<tr>
<td>mem. asser.</td>
<td>$P(c_1, c_2)$</td>
<td>supvsdBy(bob, ann)</td>
<td>$(c_1^I, c_2^I) \in P^I$</td>
</tr>
</tbody>
</table>

*Note:* We make the **unique-name assumption**, i.e., different constants denote different objects.
## DL-Lite captures conceptual modeling formalisms

<table>
<thead>
<tr>
<th>Modeling construct</th>
<th>DL-Lite</th>
<th>FOL formalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA on classes</td>
<td>$A_1 \sqsubseteq A_2$</td>
<td>$\forall x (A_1(x) \rightarrow A_2(x))$</td>
</tr>
<tr>
<td>... and on relations</td>
<td>$R_1 \sqsubseteq R_2$</td>
<td>$\forall x, y (R_1(x, y) \rightarrow R_2(x, y))$</td>
</tr>
<tr>
<td>Disjointness of classes</td>
<td>$A_1 \sqsubseteq \neg A_2$</td>
<td>$\forall x (A_1(x) \rightarrow \neg A_2(x))$</td>
</tr>
<tr>
<td>... and of relations</td>
<td>$R_1 \sqsubseteq \neg R_2$</td>
<td>$\forall x, y (R_1(x, y) \rightarrow \neg R_2(x, y))$</td>
</tr>
<tr>
<td>Domain of relations</td>
<td>$\exists P \sqsubseteq A_1$</td>
<td>$\forall x (\exists y (P(x, y)) \rightarrow A_1(x))$</td>
</tr>
<tr>
<td>Range of relations</td>
<td>$\exists P^- \sqsubseteq A_2$</td>
<td>$\forall x (\exists y (P(y, x)) \rightarrow A_2(x))$</td>
</tr>
<tr>
<td>Mandatory participation (min card = 1)</td>
<td>$A_1 \sqsubseteq \exists P$</td>
<td>$\forall x (A_1(x) \rightarrow \exists y (P(x, y)))$</td>
</tr>
<tr>
<td></td>
<td>$A_2 \sqsubseteq \exists P^-$</td>
<td>$\forall x (A_2(x) \rightarrow \exists y (P(y, x)))$</td>
</tr>
<tr>
<td>Functionality (max card = 1)</td>
<td>(funct $P$)</td>
<td>$\forall x, y, y' (P(x, y) \land P(x, y') \rightarrow y = y')$</td>
</tr>
<tr>
<td></td>
<td>(funct $P^-$)</td>
<td>$\forall x, x', y (P(x, y) \land P(x', y) \rightarrow x = x')$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Capturing UML class diagrams/ER schemas in \textit{DL-Lite}

\begin{itemize}
\item Manager $\sqsubseteq$ Researcher
\item Princlnv $\sqsubseteq$ Manager
\item Coordinator $\sqsubseteq$ Manager
\item Princlnv $\sqsubseteq$ $\neg$Coordinator
\item Researcher $\sqsubseteq$ $\exists$salary
\item $\exists$salary$^-$ $\sqsubseteq$ xsd:int
\item (\textit{funct} salary)
\item $\exists$worksFor $\sqsubseteq$ Researcher
\item $\exists$worksFor$^-$ $\sqsubseteq$ Project
\item Researcher $\sqsubseteq$ $\exists$worksFor
\item Project $\sqsubseteq$ $\exists$worksFor$^-$
\item $\exists$manages $\sqsubseteq$ Coordinator
\item $\exists$manages$^-$ $\sqsubseteq$ Project
\item Coordinator $\sqsubseteq$ $\exists$manages
\item Project $\sqsubseteq$ $\exists$manages$^-$
\item manages $\sqsubseteq$ worksFor
\item (\textit{funct} manages)
\item (\textit{funct} manages$^-$)
\end{itemize}

\textit{Note}: \textit{DL-Lite} cannot capture completeness of a hierarchy. This would require \textit{disjunction} (i.e., OR).
Query answering via \textbf{query rewriting}

Given a (U)CQ $q$ and an ontology $O = \langle T, A \rangle$:  

1. Compute the perfect rewriting of $q$ w.r.t. $T$, which is a FOL query.
2. Evaluate the perfect rewriting over $A$. (Ignore the mapping for now.)

To compute the perfect rewriting, starting from $q$, iteratively get a CQ $q'$ to be processed and either:

- expand an atom of $q'$ using an inclusion axiom, or
- unify atoms in $q'$ to obtain a more specific CQ to be further expanded.

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

We can restrict expansion and unification so as to ensure termination without losing completeness [C. \textit{et al.}, 2007a].

\textbf{Note}: negative inclusions and functionalities play a role in ontology satisfiability, but can be ignored during query answering (i.e., we have \textit{separability}).
Query answering in *DL-Lite – Example*

**TBox:**

- Coordinator ⊑ Researcher \( \forall x(\text{Coordinator}(x) \rightarrow \text{Researcher}(x)) \)
- Researcher ⊑ ∃worksFor \( \forall x(\text{Researcher}(x) \rightarrow \exists y(\text{worksFor}(x, y))) \)
- ∃worksFor\(\neg\) ⊑ Project \( \forall x(\exists y(\text{worksFor}(y, x)) \rightarrow \text{Project}(x)) \)

**Query:** \( q(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y) \)

**Perfect rewriting:**

- \( q(x) \leftarrow \text{worksFor}(x, y), \text{Project}(y) \)
- \( q(x) \leftarrow \text{worksFor}(x, y), \text{worksFor}(\_ , y) \)
- \( q(x) \leftarrow \text{worksFor}(x, \_ ) \)
- \( q(x) \leftarrow \text{Researcher}(x) \)
- \( q(x) \leftarrow \text{Coordinator}(x) \)

**ABox:**

- \( \text{worksFor}(\text{serge}, \text{webdam}) \)
- \( \text{worksFor}(\text{georg}, \text{diadem}) \)
- \( \text{Coordinator}(\text{serge}) \)
- \( \text{Coordinator}(\text{marie}) \)

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer \{serge, georg, marie\}.
Complexity of query answering in \textit{DL-Lite}

Ontology satisfiability and all classical DL reasoning tasks are:

- Efficiently tractable in the size of the TBox (i.e., \textit{PTIME}).
- Very efficiently tractable in the size of the ABox (i.e., \textit{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:

- \textit{PTIME} in the size of the TBox.
- \textit{AC^0} in the size of the ABox.
- Exponential in the size of the query, more precisely \textit{NP-complete}.

In theory this is not bad, since this is precisely the complexity of evaluating CQs in plain relational DBs.
Tracing the expressivity boundary

<table>
<thead>
<tr>
<th>Lhs concept</th>
<th>Rhs concept</th>
<th>funct.</th>
<th>Relation incl.</th>
<th>Data complexity of query answering</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DL$-$Lite$</td>
<td>$\exists P.A$</td>
<td>$\sqrt{*}$</td>
<td>$\sqrt{*}$</td>
<td>in $AC^0$</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P.A$</td>
<td>$A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
</tr>
<tr>
<td>$A</td>
<td>\forall P.A$</td>
<td>$A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P.A$</td>
<td>$\sqrt$</td>
<td>$\neg -$</td>
<td>NLogSpace-hard</td>
</tr>
<tr>
<td>$A</td>
<td>\forall P.A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
<td>PTime-hard</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P.A</td>
<td>A_1 \sqcap A_2$</td>
<td>$A$</td>
<td>$\neg -$</td>
</tr>
<tr>
<td>$A</td>
<td>\forall P.A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
<td>PTime-hard</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P.A</td>
<td>\exists P^- .A$</td>
<td>$A</td>
<td>\exists P$</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P</td>
<td>\exists P^- $</td>
<td>$A</td>
<td>\exists P</td>
</tr>
<tr>
<td>$A</td>
<td>\neg A$</td>
<td>$A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
</tr>
<tr>
<td>$A</td>
<td>A_1 \sqcup A_2$</td>
<td>$A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
</tr>
<tr>
<td>$A</td>
<td>\forall P.A$</td>
<td>$A$</td>
<td>$\neg -$</td>
<td>$\neg -$</td>
</tr>
</tbody>
</table>

From [C. et al., 2006; Artale et al., 2009].

**Notes:**

- Data complexity beyond $AC^0$ means that query answering in **not FOL rewritable**, hence cannot be delegated to a relational DBMS.
- These results pose strict bounds on the expressive power of the ontology language that can be used in OBDA.
1. Ontology-based data access framework
2. Mapping the data to the ontology
3. Query answering in OBDA
4. Ontology languages for OBDA
5. Optimizing OBDA
6. Conclusions
Experimentations and experiences

Several experimentations:

- Monte dei Paschi di Siena (led by Sapienza Univ. of Rome)
- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Horizontal Gene Transfer data and ontology
- Stanford’s “Resource Index” comprising 200 ontologies from BioPortal
- Experiments on artificial data ongoing

Observations:

- Approach highly effective for bridging impedance mismatch.
- Rewriting technique effective against incompleteness in the data.

However, performance is a major issue that still prevents large-scale deployment of this technology.
Size of the rewriting

Example

TBox \( \mathcal{T} \):

\[
q(x) \leftarrow A(x), P(x, y), A(y), P(y, z), A(z)
\]

UCQ rewriting of \( q \) w.r.t. \( \mathcal{T} \) contains 729 CQs i.e., a UNION of 729 SPJ SQL queries

The size of UCQ rewritings may become very large

- In the worst case, it may be \( O((|\mathcal{T}| \cdot |q|)^{|q|}) \), i.e., exponential in \( |q| \).
- Unfortunately, this blowup occurs also in practice.
Taming the size of the rewriting

Note: It is not possible to avoid rewriting altogether, since this would require in general to materialize an infinite database [C. et al., 2007a].

Several techniques have been proposed recently to limit the size of the rewriting:

- Alternative rewriting techniques [Pérez-Urbina et al., 2010]: more efficient algorithm based on resolution, but produces still an exponential UCQ.
- Combined approach [Kontchakov et al., 2010]: combines partial materialization with rewriting:
  - When $\mathcal{T}$ contains no role inclusions rewriting is polynomial.
  - But in general rewriting is exponential.
  - Materialization requires control over the data sources and might not be applicable in an OBDA setting.
- Rewriting into non-recursive Datalog:
  - Presto system [Rosati and Almatelli, 2010]: still worst-case exponential.
  - Polynomial rewriting for Datalog$^{\pm}$ [Gottlob and Schwentick, 2012]: rewriting uses polynomially many new existential variables and “guesses” a relevant portion of the canonical model for the TBox.

See [Kikot et al., 2012a; Kikot et al., 2012b] for discussion and further results.
A holistic approach to optimization

Recall our main objective

Given an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a set of queries, compute the certain answers of such queries w.r.t. $\mathcal{O}$ as efficiently as possible.

Observe:

- The size of the rewriting is only one coordinate in the problem space.
- Optimizing rewriting is necessary but not sufficient, since the more compact rewritings are in general much more difficult to evaluate.
- In fact, the efficiency of the query evaluation by the DBMS is the crucial factor.

Hence, a holistic approach is required, that considers all components of an OBDA system, i.e.:

- the TBox $\mathcal{T}$,
- the mappings $\mathcal{M}$,
- the data sources $\mathcal{S}$ with their dependencies, and
- the query load.
(Virtual) ABox dependencies

As in databases, we can exploit **dependencies on the data and on the ABox** to optimize query processing.

ABox dependencies express **conditions on the data in the (virtual) ABox**.

- **Syntax:** \( C_1 \preceq C_2 \) \( \quad R_1 \preceq R_2 \)
- **Meaning:** constrain the assertions present in the ABox
  - \( A \models A_1 \preceq A_2 \) iff \( A_1(d) \in A \) implies \( A_2(d) \in A \)
  - \( A \models A \preceq \exists P \) iff \( A(d) \in A \) implies \( P(d, d') \in A \) for some \( d' \)
  - \( A \models P_1 \preceq P_2 \) iff \( P_1(d, d') \in A \) implies \( P_2(d, d') \in A \)
  - \( \ldots \)

**Note:** ABox dependencies are fundamentally different from TBox assertions. They behave similarly to database constraints, since they constrain the ABox itself, and not the models.
Exploiting ABox/data dependencies in OBDA

In an OBDA system, ABox/data dependencies have an impact that spans all system components:

- Dependencies on the sources $S$ induce via the mapping $\mathcal{M}$ dependencies on the virtual ABox $\mathcal{V} = \mathcal{M}(S)$.
- The mapping itself in general induces additional dependencies on $\mathcal{V}$ (that do not directly depend on $S$).
- Dependencies on $\mathcal{V}$ interact with the TBox $\mathcal{T}$, and such interaction can be exploited for optimization.
OBDA optimization techniques

We exploit data dependencies in the following optimization techniques:

1. **Eliminating redundant TBox assertions**
   [Rodriguez-Muro and C., 2012].

2. **Semantic Index to efficiently deal with concept hierarchies**
   [Rodriguez-Muro and C., 2012].

3. **Optimize query rewriting algorithm so as to eliminate rewritings that are redundant w.r.t. query containment under dependencies**
   [Rodríguez-Muro and C., 2011; Rosati, 2012].
Eliminating redundant TBox assertions

TBox optimization is based on a characterization of assertions in a TBox $\mathcal{T}$ that are redundant wrt a set $\Sigma$ of ABox dependencies.

**Example (Direct redundancy)**

Let $\mathcal{T}$ be:

```
  Person ⊑ Human
  ∃hasFather
```

Let $\Sigma$ be:

```
  Person ⊑ Human
  ∃hasFather
```

Note: $\Sigma$ may enforce e.g., that $\text{hasFather}(\text{luisa}, \text{franz}) \in \mathcal{A}$ implies $\text{Human}(\text{luisa}) \in \mathcal{A}$.

Then $\text{Person} \sqsubseteq \text{Human}$ is **redundant** in $\mathcal{T}$.

The overall characterization of redundant TBox assertions is more involved (see [Rodriguez-Muro and C., 2012]).
Computing an optimized TBox

Given a TBox $\mathcal{T}$ and a set $\Sigma$ of ABox dependencies:

1. Compute the deductive closure $\mathcal{T}_{cl}$ of $\mathcal{T}$ (at most quadratic in size of $\mathcal{T}$).
2. Compute the deductive closure $\Sigma_{cl}$ of $\Sigma$ (at most quadratic in size of $\Sigma$).
3. Eliminate from $\mathcal{T}_{cl}$ all TBox assertions redundant wrt $\Sigma_{cl}$, obtaining $\mathcal{T}_{opt}$.

**Notes:**

- $\mathcal{T}_{opt}$ can be computed in polynomial time in the size of $\mathcal{T}$ and $\Sigma$.
- $\mathcal{T}_{opt}$ might be much smaller than $\mathcal{T}$.

Query answering using the optimized TBox

For every (virtual) ABox $\mathcal{A}$ satisfying $\Sigma$ and for every UCQ $q$, we have that

$$\text{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \text{cert}(q, \langle \mathcal{T}_{opt}, \mathcal{A} \rangle).$$

Hence, $\mathcal{T}_{opt}$ can be used instead of $\mathcal{T}$ independently of the adopted query rewriting method (provided the ABox satisfies $\Sigma$).
Deriving ABox dependencies

Dependencies on $\mathcal{M}(S)$ may be derived directly from dependencies on the data in $S$:

**Example**

Suppose we have two tables in $S$:

- ResTable[$SSN$: String, ...] stores data about researchers
- ManTable[$SSN$: String, ...] stores data about managers

Consider the following mapping $\mathcal{M}$:

$m_1$: SELECT $SSN$ FROM ResTable $\leadsto$ Researcher($person(SSN)$)
$m_2$: SELECT $SSN$ FROM ManTable $\leadsto$ Manager($person(SSN)$)

If $S$ satisfies the inclusion dependency ManTable[$SSN$] $\subseteq$ ResTable[$SSN$], then $\mathcal{M}(S)$ satisfies the dependency Manager $\preceq$ Researcher.
Deriving ABox dependencies (cont’d)

Induced by the form of the data in $S$ and the mapping $\mathcal{M}$:

Example

Suppose in $S$ we have one table: $\text{ResTable}[SSN: \text{String}, \text{IsMan}: \text{Boolean}, \ldots]$  
- Stores data about researchers (including managers).
- For managers the value of $\text{IsMan}$ is true, otherwise it is false.

Consider the following mapping $\mathcal{M}$:

- $m_1$: SELECT SSN FROM ResTable $\leadsto$ Researcher($\text{person}(SSN)$)  
- $m_2$: SELECT SSN FROM ResTable WHERE IsMan='true' $\leadsto$ Manager($\text{person}(SSN)$)

We have that $\mathcal{M}(S)$ satisfies the dependency $\text{Manager} \preceq \text{Researcher}$. This holds since the lhs query of $m_2$ is contained in the lhs query of $m_1$ (in the traditional sense of query containment in DBs).

This situation corresponds to a natural way of constructing mappings, and is very common in OBDA systems.
Deriving ABox dependencies (cont’d)

Mappings may enforce dependencies “corresponding” to TBox assertions:

Example

Suppose that in $S$ we have one table: $\text{ResTable}[\text{SSN}: \text{String}, \text{Type}: \text{Char}, \ldots ]$

- Stores data about researchers of all types.
- The value of $\text{Type}$ encodes the type or researcher: ’m’ for managers, ’p’ for principal investigators, ’c’ for coordinators, and ’o’ for other researchers.

Suppose that $T$ includes $\text{PrincInv} \sqsubseteq \text{Manager}$.

We can define a mapping $\mathcal{M}$ that induces suitable dependencies:

$m_1$: SELECT SSN FROM ResTable $\leadsto$ $\text{Researcher(person(SSN))}$

$m_2$: SELECT SSN FROM ResTable WHERE Type=’p’ $\leadsto$ $\text{PrincInv(person(SSN))}$

$m_3$: SELECT SSN FROM ResTable WHERE Type=’c’ $\leadsto$ $\text{Coordinator(person(SSN))}$

$m_4$: SELECT SSN FROM ResTable WHERE Type=’m’ OR Type=’p’ OR Type=’c’ $\leadsto$ $\text{Manager(person(SSN))}$

We have that $\mathcal{M}(S)$ satisfies the dependency $\text{PrincInv} \sqsubseteq \text{Manager}$. 
Semantic Index

Storage technique that allows for efficient querying of (concept) hierarchies.

Example

TBox $\mathcal{T}$:

$A \prec B \prec C \prec D \prec E \prec F \prec G \prec H \prec I$

$A \{1, 8\}$

$B \{2, 5\}$

$C \{6, 9\}$

$D \{3, 3\}$

$E \{4, 4\}$

$F \{5, 5\}$

$G \{7, 7\}$

$H \{8, 8\}$

$I \{9, 9\}$

ABox $\mathcal{A} = \{B(v), E(w), H(v)\}$

Stored as:

<table>
<thead>
<tr>
<th>Obj</th>
<th>Idx</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>2</td>
</tr>
<tr>
<td>$w$</td>
<td>4</td>
</tr>
<tr>
<td>$v$</td>
<td>8</td>
</tr>
</tbody>
</table>

1. We associate to each concept in $\mathcal{T}$ a numeric index, essentially by doing a breadth-first visit of the concept hierarchy.

2. We associate to each concept a (set of) interval(s), such that the interval(s) of a concept covers the indexes of all its subconcepts.

3. As ABox, we maintain a single table for all concepts, which associates to each individual $c$ the numeric indexes of the concepts of which $c$ is an instance.

Note: Similar ideas have been applied previously in different contexts [Agrawal et al., 1989; DeHaan et al., 2003].
Queries over concept hierarchies can now be expressed as simple range queries.

Example

TBox $\mathcal{T}$:

- $A$: [1, 8]
- $B$: [2, 5]
- $C$: [6, 9]
- $D$: [3, 3]
- $E$: [4, 4]
- $F$: [5, 5]
- $G$: [7, 7]
- $H$: [8, 8]
- $I$: [9, 9]

ABox $\mathcal{A} = \{B(v), E(w), H(v)\}$

Stored as:

<table>
<thead>
<tr>
<th>Obj</th>
<th>Idx</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>2</td>
</tr>
<tr>
<td>$w$</td>
<td>4</td>
</tr>
<tr>
<td>$v$</td>
<td>8</td>
</tr>
</tbody>
</table>

For example, to obtain the instances of concept $B$, we can pose the query:

```sql
SELECT Obj FROM ConceptTable WHERE Idx >= 2 AND Idx <= 5
```

- We can construct a Semantic Index in polynomial time in the size of $\mathcal{T}$.
- We can also use the range queries associated to concepts to define the mappings from the data to the ontology.
- In this way, data sources maintaining objects organized in large hierarchies can be queried very efficiently.
Experimentation using Stanford’s “Resource Index”

- Semantic search over textual documents annotated with concepts from bio-ontologies.
- 200 ontologies from Bio-portal:
  - ontology language is very simple: only concept hierarchies
  - very large ontology: 200K concepts, millions of subclass relations
- Current system uses forward chaining:
  - Naive chase: 7 days
  - Optimized chase: 40 mins
  - Storage cost: 16GB base data + 70GB of derived data
  - Split second responses
- Pure query rewriting based approaches cannot cope with the size of the ontology.
- Semantic index-based rewriting:
  - DAG computation and indexing: 5 mins
  - Cost: 16 GB (no additional storage needed)
  - Rewriting reduces to a single query, split second responses
<table>
<thead>
<tr>
<th>Outline</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ontology-based data access framework</td>
<td></td>
</tr>
<tr>
<td>2. Mapping the data to the ontology</td>
<td></td>
</tr>
<tr>
<td>3. Query answering in OBDA</td>
<td></td>
</tr>
<tr>
<td>4. Ontology languages for OBDA</td>
<td></td>
</tr>
<tr>
<td>5. Optimizing OBDA</td>
<td></td>
</tr>
<tr>
<td>6. Conclusions</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

- Ontology-based data access are challenging problems with great practical relevance.

- In this setting, the size of the data is a critical parameter that must guide technological choices.

- Theoretical foundations provide a solid basis for system development.

- Practical deployment of this technology in real world scenarios is ongoing, but requires further research.

- Adoption of a holistic approach, considering all components of OBDA systems seems the only way to cope with real-world challenges.
Further research directions

- Extensions of the ontology languages, e.g., towards $n$-ary relations [Calì et al., 2009; Baget et al., 2011; Gottlob and Schwentick, 2012].

- Dealing with inconsistency in the ontology.

- Ontology-based update.

- Coping with evolution of data in the presence of ontological constraints.

- Dealing with different kinds of data, besides relational sources: XML, graph-structured data, RDF and linked data.

- Close connection to work carried out in the Semantic Web on Triple Stores.
OBDA framework developed at the Free Univ. of Bozen-Bolzano.

http://ontop.inf.unibz.it/

Optique use cases

Statoil Exploration

- OBDA technologies developed in Optique could free as much as 30% of experts’ time.
- Potential savings: €70,000,000 per year

Siemens Energy Services

- OBDA technologies developed in Optique could reduce data acquisition time by at least 25%.
- Potential savings: €1,000,000/year/service-center
  \[= €50,000,000/year\]
Thanks

Thanks to the many people that contributed to this work:

- Alessandro Artale
- Elena Botoeva
- Giuseppe De Giacomo
- Roman Kontchakov
- Domenico Lembo
- Maurizio Lenzerini
- Antonella Poggi
- Mariano Rodriguez Muro
- Riccardo Rosati
- Michael Zakharyaschev
- many students
References I

Efficient management of transitive relationships in large data and knowledge bases.

The DL-Lite family and relations.

On rules with existential variables: Walking the decidability line.
Artificial Intelligence, 175(9–10):1620–1654, 2011.

[Berardi et al., 2005] Daniela Berardi, Diego C., and Giuseppe De Giacomo.
Reasoning on UML class diagrams.

On taxonomic reasoning in conceptual design.
References II


Conceptual modeling with description logics.


Description logics in data management.


On the decidability of query containment under constraints.


Unifying class-based representation formalisms.


*DL-Lite*: Practical reasoning for rich DLs.


Tailoring OWL for data intensive ontologies.


*DL-Lite*: Tractable description logics for ontologies.

References IV


Data complexity of query answering in description logics.


Tractable reasoning and efficient query answering in description logics: The DL-Lite family.


Actions and programs over description logic ontologies.

[C. et al., 2007c] Diego C., Thomas Eiter, and Magdalena Ortiz.

Answering regular path queries in expressive description logics: An automata-theoretic approach.

Data integration through $DL$-$Lite_A$ ontologies.


Path-based identification constraints in description logics.


Conjunctive query containment and answering under description logics constraints.


[Calì et al., 2009] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz.

Datalog$^{±}$: a unified approach to ontologies and integrity constraints.

In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 14–30, 2009.
References VI

[DeHaan et al., 2003] David DeHaan, David Toman, Mariano P. Consens, and M. Tamer Özsu.
A comprehensive XQuery to SQL translation using dynamic interval encoding.

Query answering in the description logic Horn-$SHIQ$.

[Eiter et al., 2009] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus.
Query answering in description logics with transitive roles.

Conjunctive query answering for the description logic $SHIQ$.
[Glimm et al., 2008b] Birte Glimm, Ian Horrocks, and Ulrike Sattler.
   Unions of conjunctive queries in $SHOQ$.

   Rewriting ontological queries into small nonrecursive Datalog programs.

   Long rewritings, short rewritings.

   Conjunctive query answering with OWL 2 QL.
References


The combined approach to query answering in DL-Lite.


On the satisfiability of dependency constraints in entity-relationship schemata.


Combining Horn rules and description logics in CARIN.


The complexity of conjunctive query answering in expressive description logics.

[Ortiz et al., 2006] Maria Magdalena Ortiz, Diego C., and Thomas Eiter.
Characterizing data complexity for conjunctive query answering in expressive description logics.

[Ortiz et al., 2008] Magdalena Ortiz, Diego C., and Thomas Eiter.
Data complexity of query answering in expressive description logics via tableaux.

[Pérez-Urbina et al., 2010] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks.
Tractable query answering and rewriting under description logic constraints.

Linking data to ontologies.

[Queralt et al., 2012] Anna Queralt, Alessandro Artale, Diego C., and Ernest Teniente.
OCL-Lite: Finite reasoning on UML/OCL conceptual schemas.
[Rodríguez-Muro and C., 2011] Mariano Rodríguez-Muro and Diego C.
Dependencies to optimize ontology based data access.

High performance query answering over *DL-Lite* ontologies.

[Rodríguez-Muro, 2010] Mariano Rodríguez-Muro.
*Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics.*

Improving query answering over *DL-Lite* ontologies.

Query rewriting under extensional constraints in DL-Lite.