Scalable End-User Access to Big Data

Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

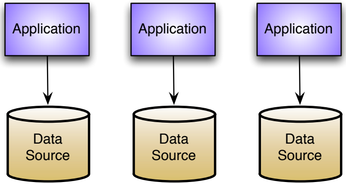
Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen-Bolzano

16th International Conference on Artificial Intelligence:
Methodology, Systems, Applications

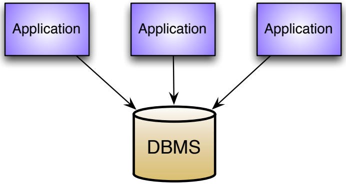
Varna, Bulgaria
11 September 2014

Data management in information systems

Pre-DBMS architecture:

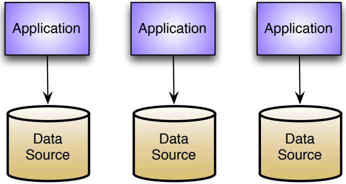


Ideal architecture based on a DBMS:



Data management today

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



Data is:

- heterogeneous
- distributed
- redundant or even duplicated
- incoherent

unibz

How to find the "right" data?

Simple case

Complex case

Challenges:

- Effort spent on actually finding and extracting the data.
  $\leadsto$ Creativity and ability to explore are hampered.
- Adding new data sources is painful.

Example: 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

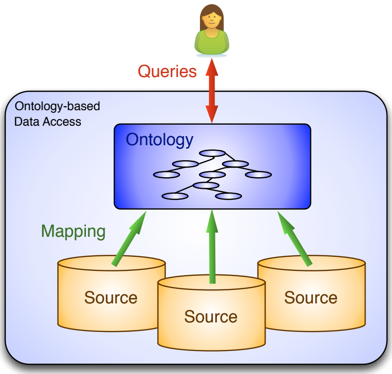
Solution proposed in the Optique project

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

- 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.

The OBDA approach is based on **formalisms grounded in logic**, with well understood semantics and computational properties.

Ontology-based data access: Architecture



An OBDA 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 OBDA specification is a triple $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{T}$ is the intensional level of an ontology.
  We consider ontologies formalized in description logics (DLs), hence the intensional level is a DL TBox.
- $\mathcal{S}$ is a (federated) relational database schema for the data sources, possibly with constraints;
- $\mathcal{M}$ is a set of mapping assertions, each one of the form

$$\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$$

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

An OBDA system is a pair $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, where

- $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is an OBDA specification, and
- $\mathcal{D}$ is a relational database compliant with $\mathcal{S}$.

Ontology-based data access: Semantics

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation of the TBox $\mathcal{T}$.

Semantics of an OBDA system

$\mathcal{I}$ is a **model** of $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, with $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ if:

- $\mathcal{I}$ is a FOL model of $\mathcal{T}$, and
- $\mathcal{I}$ satisfies $\mathcal{M}$ w.r.t. $\mathcal{D}$, i.e., it satisfies every assertion in $\mathcal{M}$ w.r.t. $\mathcal{D}$.

Semantics of mappings

We say that $\mathcal{I}$ **satisfies** $\Phi(\vec{x}) \leadsto \Psi(\vec{x})$ w.r.t. a database $\mathcal{D}$, if the FOL sentence

$$\forall \vec{x}.\, \Phi(\vec{x}) \rightarrow \Psi(\vec{x})$$

is true in $\mathcal{I} \cup \mathcal{D}$.

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

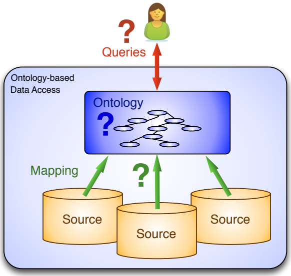
unibz

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**?



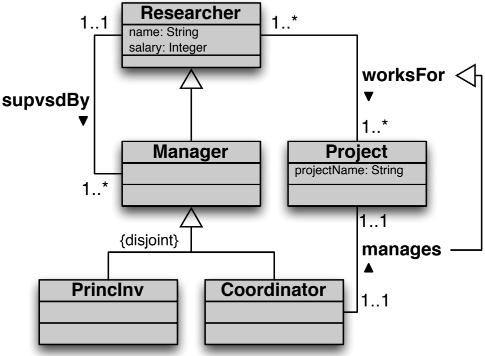
The choices that we make have to take into account the tradeoff between expressive power and efficiency of inference/query answering.

We are in a setting where we want to access big data, so **efficiency w.r.t. the data** plays an important role.

unibz

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].



$$
\begin{array}{rcl}
\mathsf{Manager} & \sqsubseteq & \mathsf{Researcher} \\
\mathsf{PrincInv} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{Coordinator} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{PrincInv} & \sqsubseteq & \neg\mathsf{Coordinator} \\
\\
\mathsf{Researcher} & \sqsubseteq & \exists \mathit{salary} \\
\exists \mathit{salary}^- & \sqsubseteq & \mathtt{xsd{:}int} \\
 & (\textbf{funct}\ \mathit{salary}) & \\
\\
\exists \mathsf{manages} & \sqsubseteq & \mathsf{Coordinator} \\
\exists \mathsf{manages}^- & \sqsubseteq & \mathsf{Project} \\
\mathsf{Coordinator} & \sqsubseteq & \exists \mathsf{manages} \\
\mathsf{Project} & \sqsubseteq & \exists \mathsf{manages}^- \\
\mathsf{manages} & \sqsubseteq & \mathsf{worksFor} \\
 & (\textbf{funct}\ \mathsf{manages}) & \\
 & (\textbf{funct}\ \mathsf{manages}^-) & \\
 & \cdots &
\end{array}
$$

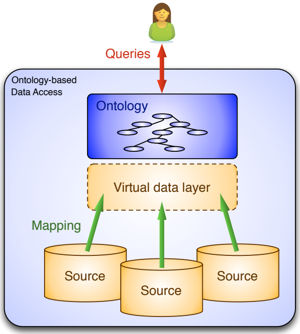
Mapping the data to the ontology

In an OBDA system $\mathcal{O} = \langle\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D}\rangle$, the **mapping** $\mathcal{M}$ encodes how the data $\mathcal{D}$ in the source(s) $\mathcal{S}$ should be used to populate the elements of $\mathcal{T}$.

Virtual data layer

The data $\mathcal{D}$ and the mapping $\mathcal{M}$ define a virtual data layer $\mathcal{V} = \mathcal{M}(\mathcal{D})$

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



unibz

Concrete mapping languages

Several proposals for concrete languages to map a relational DB to an ontology:

- They assume that the ontology is populated in terms of RDF triples.
- Some template mechanism is used to specify the triples to instantiate.

Examples: D2RQ[1], SML[2], Ontop[3]

R2RML

- Most popular RDB to RDF mapping language
- W3C Recommendation 27 Sep. 2012, `http://www.w3.org/TR/r2rml/`
- R2RML mappings are themselves expressed as RDF graphs and written in Turtle syntax.

In the following, we abstract from mappings, i.e., we assume that each concept/relation of the ontology directly corresponds to a database table.

[1] `http://d2rq.org/d2rq-language`

[2] `http://sparqlify.org/wiki/Sparqlification_mapping_language`

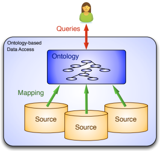
[3] `https://github.com/ontop/ontop/wiki/ObdalibObdaTurtlesyntax`

Outline

Outline

1. Ontology-based data access framework
2. Query answering in OBDA
3. Ontology languages for OBDA
4. Optimizing OBDA in Ontop
5. Conclusions

Incomplete information

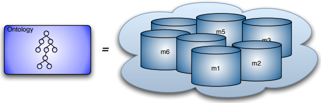


We are in a setting of **incomplete information**!!!

Incompleteness introduced:

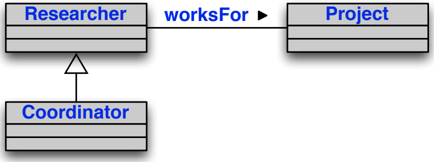
- by data source(s), in general assumed to be incomplete;
- by domain constraints encoded in the ontology.

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 1



We assume that each concept/relationship of the ontology is mapped directly to a database table.

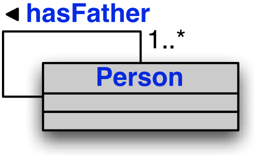
But the database tables may be **incompletely specified**, or even missing for some concepts/relationships.

DB: Coordinator $\supseteq$ { serge, marie }
Project $\supseteq$ { webdam, diadem }
worksFor $\supseteq$ { (serge,webdam), (georg,diadem) }

Query: $q(x) \leftarrow$ Researcher(x)

Answer: { serge, marie, georg }

Incomplete information – Example 2



Each person has a father, who is a person.

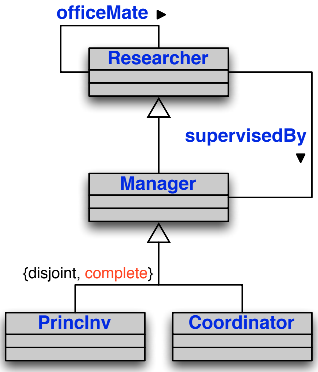
DB: Person $\supseteq$ { `john`, `nick`, `toni` }
hasFather $\supseteq$ { (`john`,`nick`), (`nick`,`toni`) }

Queries:

$$q_1(x,y) \leftarrow \mathsf{hasFather}(x,y)$$
$$q_2(x) \leftarrow \exists y.\, \mathsf{hasFather}(x,y)$$
$$q_3(x) \leftarrow \exists y_1,y_2,y_3.\, \mathsf{hasFather}(x,y_1) \wedge \mathsf{hasFather}(y_1,y_2) \wedge \mathsf{hasFather}(y_2,y_3)$$
$$q_4(x,y_3) \leftarrow \exists y_1,y_2.\, \mathsf{hasFather}(x,y_1) \wedge \mathsf{hasFather}(y_1,y_2) \wedge \mathsf{hasFather}(y_2,y_3)$$

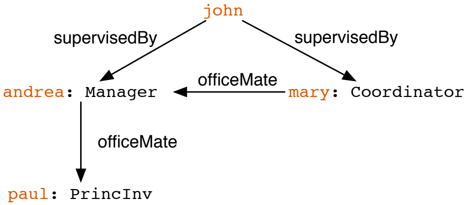
Answers: to q_1: { (`john`,`nick`), (`nick`,`toni`) }
to q_2: { `john`, `nick`, `toni` }
to q_3: { `john`, `nick`, `toni` }
to q_4: { }

QA over ontologies – Andrea's Example [4]



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

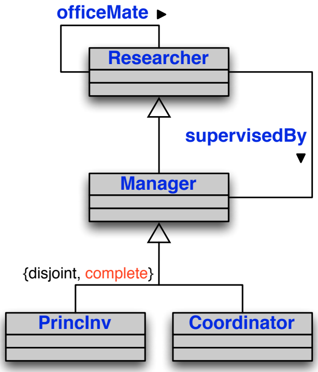
$$\begin{aligned}
\text{Researcher} &\supseteq \{\ \texttt{andrea}, \texttt{paul}, \texttt{mary}, \texttt{john}\ \}\\
\text{Manager} &\supseteq \{\ \texttt{andrea}, \texttt{paul}, \texttt{mary}\ \}\\
\text{PrincInv} &\supseteq \{\ \texttt{paul}\ \}\\
\text{Coordinator} &\supseteq \{\ \texttt{mary}\ \}\\
\text{supervisedBy} &\supseteq \{\ (\texttt{john},\texttt{andrea}), (\texttt{john},\texttt{mary})\ \}\\
\text{officeMate} &\supseteq \{\ (\texttt{mary},\texttt{andrea}), (\texttt{andrea},\texttt{paul})\ \}
\end{aligned}$$

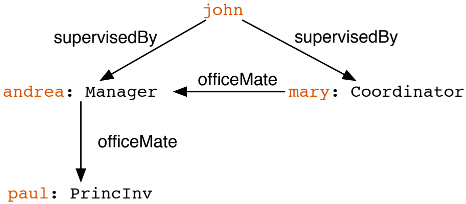


unibz

[4] By Andrea Schaerf [PhD Thesis 1994].

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





$$q(x) \leftarrow \exists y,z.\ \mathsf{supervisedBy}(x,y) \land \mathsf{Coordinator}(y) \land \mathsf{officeMate}(y,z) \land \mathsf{PrincInv}(z)$$

Answer: { john }

To obtain this answer, we need to **reason by cases**, i.e., model by model.

unibz

Query answering – Which query language to use

Certain answers, i.e., answers that are logically implied

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:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 (1)
Expressive DLs	$\geq$ 2ExpTime (2)	coNP-hard (3)

(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; C. *et al.*, 2014].

(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 OBDA with big data

Challenges

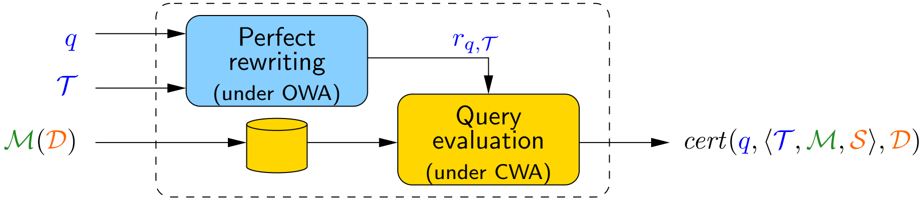
- Can we find **interesting ontology languages** for which query answering in OBDA in theory can be done efficiently (i.e., in AC^0)?
- If yes, can we answer queries in OBDA by **exploiting a relational engine**?
- If yes, can we obtain **acceptable performance in practical scenarios** involving large ontologies and big data?

Logical inference for query answering

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

$\leadsto$ Query answering by **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 $r_{q,\mathcal{T}}$ (called the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. **Query evaluation**: evaluate $r_{q,\mathcal{T}}$ over $\mathcal{M}(\mathcal{D})$ seen as a complete database (and without considering $\mathcal{T}$).
   $\rightsquigarrow$ Produces $cert(q, \langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D})$.

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

unibz

FOL-rewritability

Let:

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

$\mathcal{L}_Q$-rewritability of conjunctive query answering

Conjunctive query answering is **$\mathcal{L}_Q$-rewritable** for $\mathcal{L}_T$, 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 $\mathcal{L}_Q$.

We are especially interested in **FOL-rewritability**:

- The rewriting can be expressed in FOL, i.e., in SQL.
- Query evaluation can be delegated to a relational DBMS.

This notion was initially proposed in [C. *et al.*, 2005b; 2006; 2007a] and further intensively investigated in the KR and DB community.

Outline

1. Ontology-based data access framework
2. Query answering in OBDA
3. Ontology languages for OBDA
4. Optimizing OBDA in Ontop
5. Conclusions

unibz

Description Logics

- **Description Logics (DLs)** stem from early days (70') 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; C. *et al.*, 2013]).

unibz

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: encode terminological knowledge about the domain

Role inclusion:	$R_1 \sqsubseteq R_2$	Concept inclusion:	$C_1 \sqsubseteq C_2$
Role disjointness:	$R_1 \sqsubseteq \neg R_2$	Concept disjointness:	$C_1 \sqsubseteq \neg C_2$
Role functionality:	$(\textbf{funct}\ R)$		

ABox assertions: encode knowledge about individuals
$A(c)$, $P(c_1, c_2)$, with c_1, c_2 constants

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

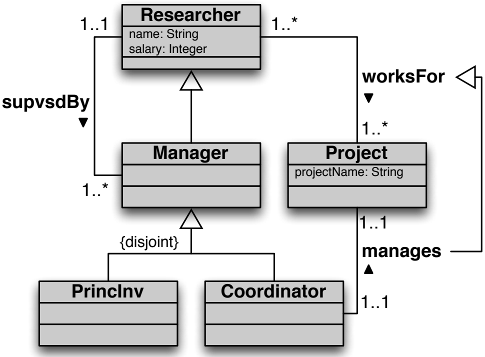
unibz

DL-Lite captures conceptual modeling formalisms

Modeling construct	*DL-Lite*	FOL formalization
ISA on classes	$A_1 \sqsubseteq A_2$	$\forall x(A_1(x) \rightarrow A_2(x))$
... and on relations	$R_1 \sqsubseteq R_2$	$\forall x, y(R_1(x,y) \rightarrow R_2(x,y))$
Disjointness of classes	$A_1 \sqsubseteq \neg A_2$	$\forall x(A_1(x) \rightarrow \neg A_2(x))$
... and of relations	$R_1 \sqsubseteq \neg R_2$	$\forall x, y(R_1(x,y) \rightarrow \neg R_2(x,y))$
Domain of relations	$\exists P \sqsubseteq A_1$	$\forall x(\exists y(P(x,y)) \rightarrow A_1(x))$
Range of relations	$\exists P^- \sqsubseteq A_2$	$\forall x(\exists y(P(y,x)) \rightarrow A_2(x))$
Mandatory participation *(min card = 1)*	$A_1 \sqsubseteq \exists P$ $A_2 \sqsubseteq \exists P^-$	$\forall x(A_1(x) \rightarrow \exists y(P(x,y)))$ $\forall x(A_2(x) \rightarrow \exists y(P(y,x)))$
Functionality *(max card = 1)*	$(\textbf{funct } P)$ $(\textbf{funct } P^-)$	$\forall x, y, y'(P(x,y) \land P(x,y') \rightarrow y = y')$ $\forall x, x', y(P(x,y) \land P(x',y) \rightarrow x = x')$
...	...	...

unibz

Capturing UML class diagrams/ER schemas in *DL-Lite*



$$
\begin{array}{rcl}
\mathsf{Manager} & \sqsubseteq & \mathsf{Researcher} \\
\mathsf{PrincInv} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{Coordinator} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{PrincInv} & \sqsubseteq & \neg\mathsf{Coordinator} \\[1ex]
\mathsf{Researcher} & \sqsubseteq & \exists \mathit{salary} \\
\exists \mathit{salary}^- & \sqsubseteq & \mathtt{xsd{:}int} \\
& (\textbf{funct}\ \mathit{salary}) & \\[1ex]
\exists \mathsf{worksFor} & \sqsubseteq & \mathsf{Researcher} \\
\exists \mathsf{worksFor}^- & \sqsubseteq & \mathsf{Project} \\
\mathsf{Researcher} & \sqsubseteq & \exists \mathsf{worksFor} \\
\mathsf{Project} & \sqsubseteq & \exists \mathsf{worksFor}^- \\[1ex]
\exists \mathsf{manages} & \sqsubseteq & \mathsf{Coordinator} \\
\exists \mathsf{manages}^- & \sqsubseteq & \mathsf{Project} \\
\mathsf{Coordinator} & \sqsubseteq & \exists \mathsf{manages} \\
\mathsf{Project} & \sqsubseteq & \exists \mathsf{manages}^- \\
\mathsf{manages} & \sqsubseteq & \mathsf{worksFor} \\
& (\textbf{funct}\ \mathsf{manages}) & \\
& (\textbf{funct}\ \mathsf{manages}^-) & \\
& \cdots &
\end{array}
$$

Note: *DL-Lite* cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

Query answering in *DL-Lite*

Query answering via query rewriting

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

1. Compute the perfect rewriting of q w.r.t. $\mathcal{T}$, which is a FOL query.
2. Evaluate the perfect rewriting over $\mathcal{A}$. (We have ignored the mapping.)

I briefly describe *PerfectRef*, a simple algorithm for Step 1 that requires to iterate over:

- rewriting steps that involve inclusion assertions, and
- unification steps.

Note: disjointness assertions and functionalities play a role in ontology satisfiability, but can be ignored during query rewriting (i.e., we have **separability**).

unibz

Query rewriting step: Basic idea

Intuition: an inclusion assertion corresponds to a logic programming rule.

Basic rewriting step:

When an atom in the query unifies with the **head** of the rule, generate a new query by substituting the atom with the **body** of the rule.

We say that the inclusion assertion **applies to** the atom.

Example

The inclusion assertion $\text{Coordinator} \sqsubseteq \text{Researcher}$

corresponds to the logic programming rule $\text{Researcher}(z) \leftarrow \text{Coordinator}(z)$.

Consider the query $\text{q}(x) \leftarrow \text{Researcher}(x)$.

By applying the inclusion assertion to the atom $\text{Researcher}(x)$, we generate:

$$\text{q}(x) \leftarrow \text{Coordinator}(x)$$

Query rewriting

To compute the perfect rewriting of a query q, start from q, iteratively get a CQ q' to be processed, and do one of the following:

- Apply to some atom of q' an inclusion assertion in $\mathcal{T}$ as follows:

$$
\begin{array}{rccc}
A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\
\exists P \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(x, \_), \ldots \\
\exists P^- \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(\_, x), \ldots \\
A \sqsubseteq \exists P & \ldots, P(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
A \sqsubseteq \exists P^- & \ldots, P(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
\exists P_1 \sqsubseteq \exists P_2 & \ldots, P_2(x, \_), \ldots & \rightsquigarrow & \ldots, P_1(x, \_), \ldots \\
P_1 \sqsubseteq P_2 & \ldots, P_2(x, y), \ldots & \rightsquigarrow & \ldots, P_1(x, y), \ldots \\
\ldots & & &
\end{array}
$$

  ('$\_$' denotes a variable that appears only once)

- Choose two atoms of q' that unify, and apply the unifier to q'.

Each time, the result of the above step is added to the queries to be processed.

Note: Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method [C. *et al.*, 2007a].

The UCQ resulting from this process is the **perfect rewriting** $r_{q,\mathcal{T}}$.

Query answering in *DL-Lite* – Example

TBox:

$$\begin{aligned} \mathsf{Coordinator} &\sqsubseteq \mathsf{Researcher} \\ \mathsf{Researcher} &\sqsubseteq \exists \mathsf{worksFor} \\ \exists \mathsf{worksFor}^- &\sqsubseteq \mathsf{Project} \end{aligned}$$

Corresponding rules:

$$\begin{aligned} \mathsf{Coordinator}(x) &\rightarrow \mathsf{Researcher}(x) \\ \mathsf{Researcher}(x) &\rightarrow \exists y(\mathsf{worksFor}(x,y)) \\ \mathsf{worksFor}(y,x) &\rightarrow \mathsf{Project}(x) \end{aligned}$$

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

Perfect rewriting:

$$\begin{aligned} \mathsf{q}(x) &\leftarrow \mathsf{worksFor}(x,y), \mathsf{Project}(y) \\ \mathsf{q}(x) &\leftarrow \mathsf{worksFor}(x,y), \mathsf{worksFor}(\_,y) \\ \mathsf{q}(x) &\leftarrow \mathsf{worksFor}(x,\_) \\ \mathsf{q}(x) &\leftarrow \mathsf{Researcher}(x) \\ \mathsf{q}(x) &\leftarrow \mathsf{Coordinator}(x) \end{aligned}$$

ABox: $\mathsf{worksFor}(\mathtt{serge}, \mathtt{webdam})$ $\mathsf{Coordinator}(\mathtt{serge})$
$\mathsf{worksFor}(\mathtt{georg}, \mathtt{diadem})$ $\mathsf{Coordinator}(\mathtt{marie})$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\mathtt{serge},\ \mathtt{georg},\ \mathtt{marie}\}$.

unibz

Complexity of query answering in *DL-Lite*

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**, more precisely 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

	Lhs concept	Rhs concept	funct.	Relation incl.	Data complexity of query answering
0	*DL-Lite*		$\surd^*$	$\surd^*$	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**

From [C. *et al.*, 2006; Artale *et al.*, 2009; C. *et al.*, 2013].

Notes:

- Data complexity beyond AC^0 means that query answering is **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.

unibz

Outline

unibz

Experimentations and experiences

Several experimentations:

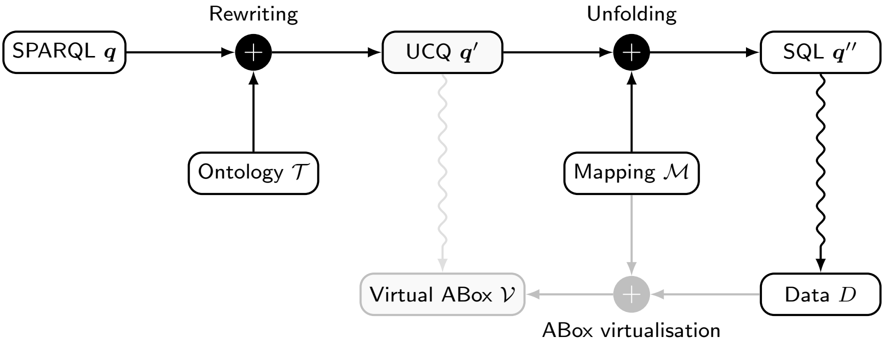
- Monte dei Paschi di Siena (led by Sapienza Univ. of Rome)
- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Horizontal Gene Transfer data and ontology
- Stanford's "Resource Index" comprising 200 ontologies from BioPortal
- Experiments on artificial data ongoing

Observations:

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

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

Query processing in a traditional OBDA system



What makes the resulting SQL query grow exponentially?

Three main factors affect the size of the resulting query q'':

Existentials: Sub-queries of q with **existentially quantified variables** might lead in general to exponentially large rewritings.

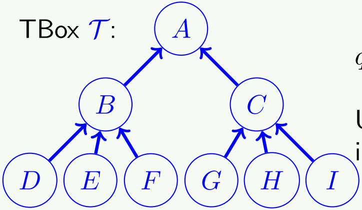
Hierarchies: Concepts / roles occurring in the query q can have **many subconcepts / subroles** according to $\mathcal{T}$, which all have to be included in the rewriting q'.

Mappings: The mapping $\mathcal{M}$ can provide **multiple definitions of the concepts and roles in the ontology**, which may result in a further exponential blowup in the unfolding step of q' to q''.

Impact of hierarchies – Example

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.*, 2012; Gottlob *et al.*, 2014] for discussion and further results.

A holistic approach to optimization

Recall our main objective

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

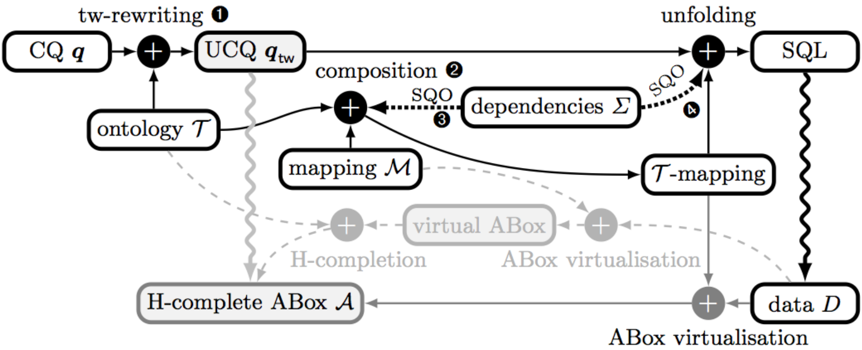
Observe:

- 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 $\mathcal{D}$ with their dependencies in $\mathcal{S}$, and
- the query load.

Optimizations in Ontop [Rodriguez-Muro *et al.*, 2013]



1. Tree-witness rewriting over H-complete ABoxes.
2. $\mathcal{T}$-mappings incorporating $\mathcal{T}$ into $\mathcal{M}$.
3. Simplification of $\mathcal{T}$-mappings using Semantic Query Optimisation (SQO).
4. Optimized unfolding.

The Ontop OBDA framework

Developed at the Free Univ. of Bozen-Bolzano: `http://ontop.inf.unibz.it/`

ontop "Stay on top of your data with semantics"

Features of Ontop

- Query language: support for SPARQL 1.0 (and part of 1.1)
- Mapping languages:
  - Intuitive Ontop mapping language
  - Support for R2RML W3C standard
- Database: Support for free and commercial DBMSs
  - PostgreSQL, MySQL, H2, DB2, ORACLE, MS SQL SERVER, TEIID, ADP
- Java library/providers for Sesame and OWLAPI
  - Sesame: a de-facto standard framework for processing RDF data
  - OWLAPI: Java API and reference implementation for OWL Ontologies
- Integrated with Protege 4.x
- Provides a SPARQL end-point (via Sesame Workbench)
- Apache open source license

Outline

Conclusions

- Ontology-based data access provides 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 with big data is ongoing, but requires further work.
- Adoption of a holistic approach, considering all components of OBDA systems seems the only way to cope with real-world challenges.

unibz

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.
- Management of mappings and ontology axioms.
- User-friendly ontology querying modalities (graphical query languages, natural language querying).

Thanks

Thank you for your attention!

. . . and thanks to many people who contributed to this work:

- Alessandro Artale (FUB)
- Elena Botoeva (FUB)
- Giuseppe De Giacomo (Uniroma1))
- Roman Kontchakov (Birkbeck)
- Davide Lanti (FUB)
- Domenico Lembo (Uniroma1)
- Maurizio Lenzerini (Uniroma1)
- Antonella Poggi (Uniroma1)
- Martin Rezk (FUB)
- Mariano Rodriguez Muro (FUB, IBM)
- Riccardo Rosati (Uniroma1)
- Guohui Xiao (FUB)
- Michael Zakharyaschev (Birkbeck)
- many students

EU IP Project

Optique

(Nov. 2012 – Oct. 2016)

unibz

References I

[Artale *et al.*, 2009] Alessandro Artale, Diego C., Roman Kontchakov, and Michael Zakharyaschev.
The *DL-Lite* family and relations.
J. of Artificial Intelligence Research, 36:1–69, 2009.

[Baget *et al.*, 2011] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat.
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.
Artificial Intelligence, 168(1–2):70–118, 2005.

[Bergamaschi and Sartori, 1992] Sonia Bergamaschi and Claudio Sartori.
On taxonomic reasoning in conceptual design.
ACM Trans. on Database Systems, 17(3):385–422, 1992.

References II

[Borgida and Brachman, 2003] Alexander Borgida and Ronald J. Brachman.
Conceptual modeling with description logics.
In Franz Baader, Diego C., Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 10, pages 349–372. Cambridge University Press, 2003.

[Borgida, 1995] Alexander Borgida.
Description logics in data management.
IEEE Trans. on Knowledge and Data Engineering, 7(5):671–682, 1995.

[C. *et al.*, 1998] Diego C., Giuseppe De Giacomo, and Maurizio Lenzerini.
On the decidability of query containment under constraints.
In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[C. *et al.*, 1999] Diego C., Maurizio Lenzerini, and Daniele Nardi.
Unifying class-based representation formalisms.
J. of Artificial Intelligence Research, 11:199–240, 1999.

References III

[C. *et al.*, 2004] Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati, and Guido 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.

[C. *et al.*, 2005a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo 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.

[C. *et al.*, 2005b] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

DL-Lite: Tractable description logics for ontologies.

In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

unibz

References IV

[C. *et al.*, 2006] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo 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.

[C. *et al.*, 2007a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

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

J. of Automated Reasoning, 39(3):385–429, 2007.

[C. *et al.*, 2007b] Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo 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.

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

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

In *Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI 2007)*, pages 391–396, 2007.

References V

[C. *et al.*, 2008a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi.
Data integration through *DL-Lite*$_{\mathcal{A}}$ ontologies.
In Klaus-Dieter Schewe and Bernhard 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.

[C. *et al.*, 2008b] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo 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.

[C. *et al.*, 2008c] Diego C., Giuseppe De Giacomo, and Maurizio Lenzerini.
Conjunctive query containment and answering under description logics constraints.
ACM Trans. on Computational Logic, 9(3):22.1–22.31, 2008.

[C. *et al.*, 2013] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
Data complexity of query answering in description logics.
Artificial Intelligence, 195:335–360, 2013.

References VI

[C. *et al.*, 2014] Diego C., Magdalena Ortiz, and Thomas Eiter.
Answering regular path queries in expressive description logics via alternating tree-automata.
Information and Computation, 237:12–55, 2014.

[Calì *et al.*, 2009] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz.
Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints.
In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 14–30, 2009.

[Eiter *et al.*, 2008] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus.
Query answering in the description logic Horn-$\mathcal{SHIQ}$.
In *Proc. of the 11th Eur. Conference on Logics in Artificial Intelligence (JELIA 2008)*, pages 166–179, 2008.

[Eiter *et al.*, 2009] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus.
Query answering in description logics with transitive roles.
In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, pages 759–764, 2009.

References VII

[Glimm *et al.*, 2008a] Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler.
Conjunctive query answering for the description logic $\mathcal{SHIQ}$.
J. of Artificial Intelligence Research, 31:151–198, 2008.

[Glimm *et al.*, 2008b] Birte Glimm, Ian Horrocks, and Ulrike Sattler.
Unions of conjunctive queries in $\mathcal{SHOQ}$.
In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 252–262, 2008.

[Gottlob and Schwentick, 2012] Georg Gottlob and Thomas Schwentick.
Rewriting ontological queries into small nonrecursive Datalog programs.
In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 254–263, 2012.

[Gottlob *et al.*, 2014] Georg Gottlob, Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, Thomas Schwentick, and Michael Zakharyaschev.
The price of query rewriting in ontology-based data access.
Artificial Intelligence, 213:42–59, 2014.

References VIII

[Kikot *et al.*, 2012] Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev.
Conjunctive query answering with OWL 2 QL.
In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 275–285, 2012.

[Kontchakov *et al.*, 2010] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev.
The combined approach to query answering in *DL-Lite*.
In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 247–257, 2010.

[Lenzerini and Nobili, 1990] Maurizio Lenzerini and Paolo Nobili.
On the satisfiability of dependency constraints in entity-relationship schemata.
Information Systems, 15(4):453–461, 1990.

[Levy and Rousset, 1998] Alon Y. Levy and Marie-Christine Rousset.
Combining Horn rules and description logics in CARIN.
Artificial Intelligence, 104(1–2):165–209, 1998.

References IX

[Lutz, 2008] Carsten Lutz.
The complexity of conjunctive query answering in expressive description logics.
In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR 2008)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 179–193. Springer, 2008.

[Ortiz *et al.*, 2006] Maria Magdalena Ortiz, Diego C., and Thomas Eiter.
Characterizing data complexity for conjunctive query answering in expressive description logics.
In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, pages 275–280, 2006.

[Ortiz *et al.*, 2008] Magdalena Ortiz, Diego C., and Thomas Eiter.
Data complexity of query answering in expressive description logics via tableaux.
J. of Automated Reasoning, 41(1):61–98, 2008.

[Pérez-Urbina *et al.*, 2010] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks.
Tractable query answering and rewriting under description logic constraints.
J. of Applied Logic, 8(2):186–209, 2010.

unibz

References X

[Poggi *et al.*, 2008] Antonella Poggi, Domenico Lembo, Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati.
Linking data to ontologies.
J. on Data Semantics, X:133–173, 2008.

[Queralt *et al.*, 2012] Anna Queralt, Alessandro Artale, Diego C., and Ernest Teniente.
OCL-Lite: Finite reasoning on UML/OCL conceptual schemas.
Data and Knowledge Engineering, 73:1–22, 2012.

[Rodriguez-Muro *et al.*, 2013] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev.
Ontology-based data access: Ontop of databases.
In *Proc. of the 12th Int. Semantic Web Conf. (ISWC)*, volume 8218 of *Lecture Notes in Computer Science*, pages 558–573. Springer, 2013.

[Rodríguez-Muro, 2010] Mariano Rodríguez-Muro.
Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics.
PhD thesis, KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, 2010.

References XI

[Rosati and Almatelli, 2010] Riccardo Rosati and Alessandro Almatelli.
Improving query answering over *DL-Lite* ontologies.
In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 290–300, 2010.

unibz