Ontologies for Data Integration

Diego Calvanese

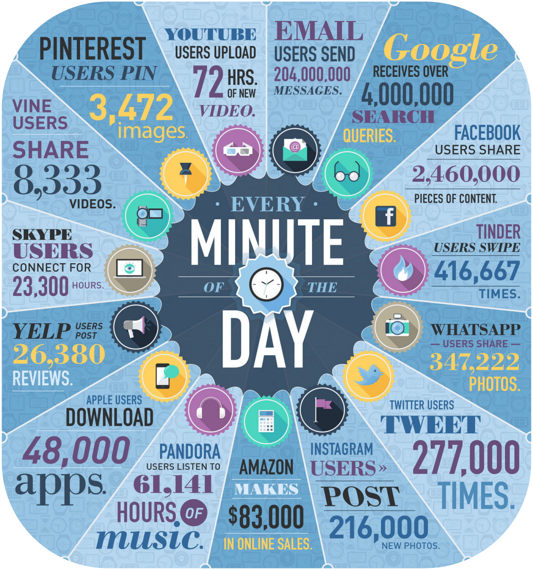
KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

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

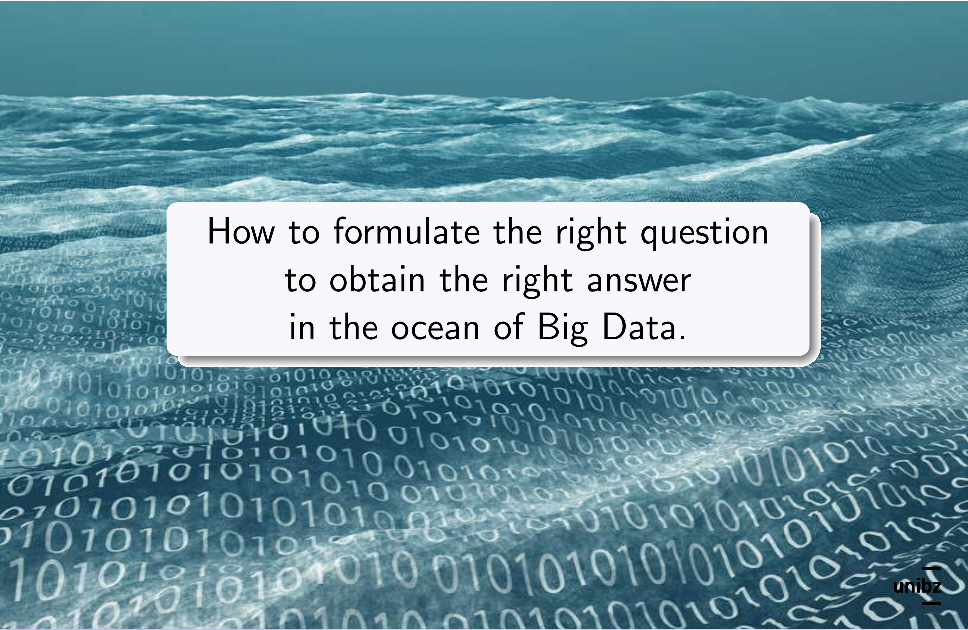
IJCAI Workshop on Formal Ontologies for Artificial Intelligence (FOfAI)

Buenos Aires, Argentina
27 July 2015

We are living in the era of Big Data



The Problem: information access



How to formulate the right question
to obtain the right answer
in the ocean of Big Data.

How much time is spent searching for data?

Engineers in industry spend a significant amount of their time searching for data that they require for their core tasks. For example, in the oil&gas industry, 30–70% of engineers' time is spent looking for data and assessing its quality (Crompton, 2008).

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 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.

How much time/money is spent searching for data?

```
SELECT [...]
FROM
db_name.table1 table1,
db_name.table2 table2a,
db_name.table2 table2b,
db_name.table3 table3a,
db_name.table3 table3b,
db_name.table3 table3c,
db_name.table3 table3d,
db_name.table4 table4a,
db_name.table4 table4b,
...
db_name.table9 table9,
db_name.table10 table10a,
db_name.table10 table10b,
db_name.table10 table10c,
db_name.table11 table11,
db_name.table12 table12,
db_name.table13 table13,
db_name.table14 table14,
db_name.table15 table15,
db_name.table16 table16
WHERE [...]
table2a.attr1='keyword' AND
table3a.attr2=table10c.attr1 AND
table3a.attr6=table6a.attr3 AND
table3a.attr9='keyword' AND
table4a.attr10 IN ('keyword') AND
table4a.attr1 IN ('keyword') AND
table5a.kinds=table4a.attr13 AND
table5b.kinds=table4c.attr74 AND
table5b.name='keyword' AND
(table6a.attr19=table10c.attr17 OR
(table6a.attr2 IS NULL AND
...
table8.attr19=table13.attr20 AND
table8.attr4='keyword' AND
table9.attr10=table16.attr11 AND
table3b.attr19=table10c.attr18 AND
table3b.attr22=table12.attr63 AND
table3b.attr66='keyword' AND
table10a.attr54=table7a.attr8 AND
table10a.attr70=table10c.attr10 AND
table10a.attr16=table4d.attr11 AND
table4c.attr99='keyword' AND
table4c.attr1='keyword' AND
table11.attr10=table5a.attr10 AND
table11.attr40='keyword' AND
table11.attr50='keyword' AND
table2b.attr1=table1.attr8 AND
table2b.attr9 IN ('keyword') AND
table2b.attr2 LIKE 'keyword'% AND
table12.attr9 IN ('keyword') AND
table7b.attr1=table2a.attr10 AND
table3c.attr13=table10c.attr1 AND
table3c.attr10=table6b.attr20 AND
table3c.attr13='keyword' AND
...
table16.attr16=table10b.attr78 AND
table16.attr5=table14.attr56 AND
table4e.attr34 IN ('keyword') AND
table4e.attr48 IN ('keyword') AND
table4f.attr89=table5b.attr7 AND
table4f.attr45 IN ('keyword') AND
table4f.attr1='keyword' AND
table10c.attr2=table4e.attr19 AND
(table10c.attr78=table12.attr56 OR
(table10c.attr55 IS NULL AND
table12.attr17 IS NULL))
```

At Statoil, it takes up to 4 days to formulate a query in SQL.

Statoil loses up to **50.000.000€** per year because of this!!

Ontologies to the rescue

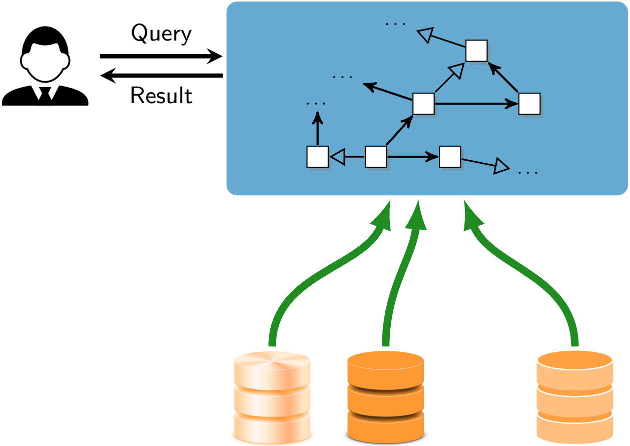
Manage data by adopting principles and techniques studied in **Knowledge Representation**.

- 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 inference services to **automatically translate the requests** into queries to the data sources.

Ontology-based data integration (OBDI)

The OBDI approach is based on **ontologies**, which are **grounded in logic**, with well understood semantics and computational properties.

Ontology-based data integration framework



Ontology
provides global vocabulary and conceptual view

Mappings
semantically link sources and ontology

Data Sources
external and heterogeneous

We achieve **logical transparency** in accessing data:

- does not know where and how the data is stored.
- can only see a conceptual view of the data.

Ontology-based data integration: Formalization

An **OBDI 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 **OBDI 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 OBDI specification, and
- $\mathcal{D}$ is a collection of relational databases compliant with $\mathcal{S}$.

Ontology-based data integration: Semantics

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

Semantics of an OBDI 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. databases $\mathcal{D}$, 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

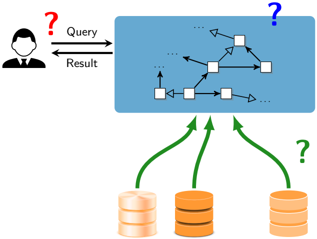
Challenges in OBDI

- How to instantiate the abstract framework?
- How to execute queries over the ontology by accessing data in the sources?
- How to deal with heterogeneity in the data?
- How to optimize performance with big data and large ontologies?
- How to address the expressivity – efficiency tradeoff?
- How to provide automated support for key tasks during design and deployment?
- How to assess the quality of the constructed system?

unibz

Instantiating the framework

1. Which is the “right” ontology language?
2. Which is the “right” query language?
3. Which is the “right” mapping language?

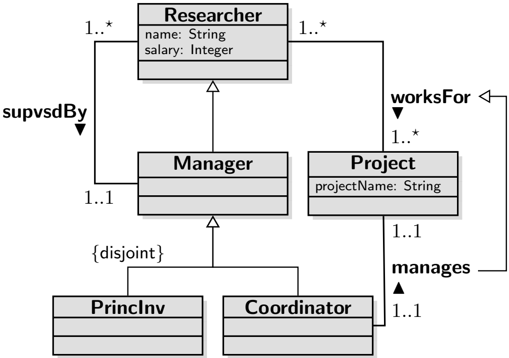


The choices that we make have to take into account the tradeoff between expressive power and efficiency of inference/query answering.

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

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].



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

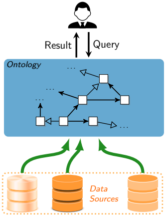
unibz

Outline

unibz

Outline

Incomplete information

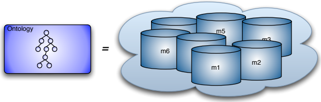


We are in a setting of **incomplete information**!!!

Incompleteness introduced:

- by data sources, 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.

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.*, 2008a; Glimm *et al.*, 2008b; 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.*, 2008b].

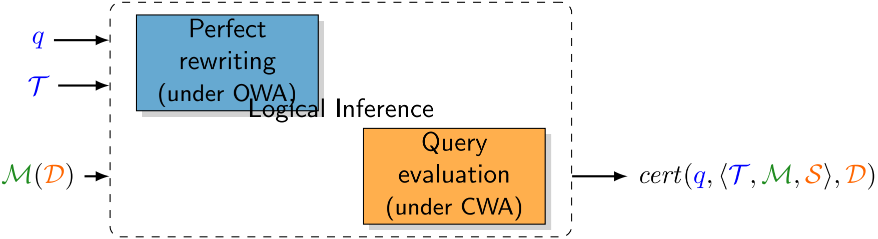
unibz

Challenges for query answering in OBDI with big data

Challenges

- Are there **interesting ontology languages** for which query answering in OBDA can be done efficiently, at least in theory (i.e., in AC^0)?
- If yes, can we answer queries in OBDA by **exploiting a relational engine** and obtain acceptable performance?
- Can we overcome limitations in the expressive power of the ontology language, by leveraging the OBDI framework?

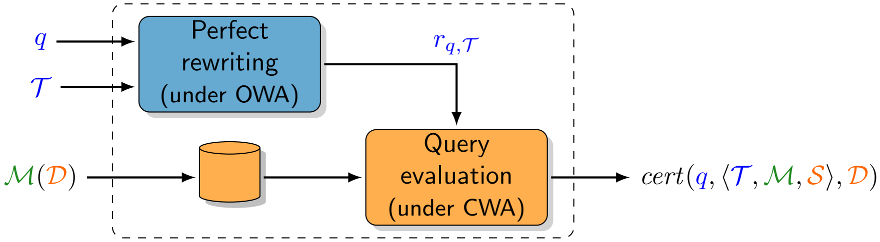
Logical inference for query answering



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 $\mathcal{O}$.

$\rightsquigarrow$ 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 $\mathcal{T}$).
   $\leadsto$ 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 $r_{q,\mathcal{T}}$.

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

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

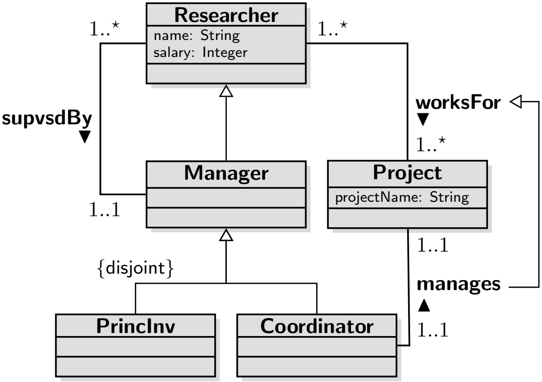
Note: *DL-Lite* distinguishes also between abstract objects and data values (ignored here).

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)*	$(\textbf{funct}\ P)$ $(\textbf{funct}\ P^-)$	$\forall x, y, y'(P(x, y) \land P(x, y') \rightarrow y = y')$ $\forall x, x', y(P(x, y) \land P(x', y) \rightarrow x = x')$
. . .	. . .	. . .

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



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

DL-Lite **cannot capture covering** constraints.
To do so, would require **disjunction**.

unibz

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** $r_{q,\mathcal{T}}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$\mathsf{Coordinator} \sqsubseteq \mathsf{Researcher}$
$\mathsf{Researcher} \sqsubseteq \exists\mathsf{worksFor}$
$\exists\mathsf{worksFor}^- \sqsubseteq \mathsf{Project}$

Corresponding rules:

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

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

Perfect rewriting:
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{Project}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, y), \mathsf{worksFor}(\_, y)$
$\mathsf{q}(x) \leftarrow \mathsf{worksFor}(x, \_)$
$\mathsf{q}(x) \leftarrow \mathsf{Researcher}(x)$
$\mathsf{q}(x) \leftarrow \mathsf{Coordinator}(x)$

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 $\{\mathtt{serge},\ \mathtt{georg},\ \mathtt{marie}\}$.

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.

Note: In the following, in line with the Semantic Web standards, we will consider CQs expressed in SPARQL, and ontology reasoning is done according to the SPARQL entailment regimes.

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.

Outline

unibz

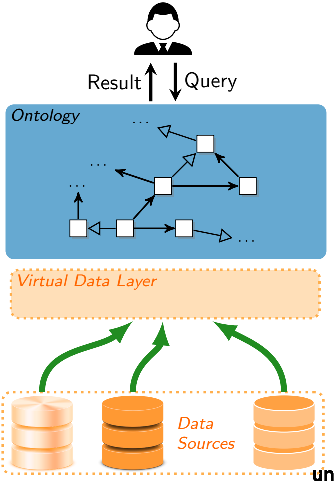
Virtual data layer

In an OBDI 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 sources $\mathcal{S}$ 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

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.

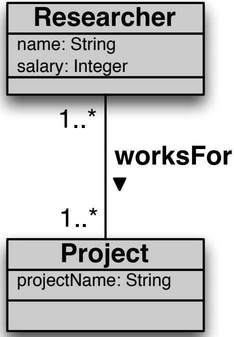
Proposed solution:

- 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:

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

- Techniques from partial evaluation of logic programs are adapted for unfolding queries over $\mathcal{T}$, by using $\mathcal{M}$, into queries over $\mathcal{S}$.

Impedance mismatch – Example



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

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

$D_2[Code\colon \text{String}, Salary\colon \text{Int}]$
Researchers' code with salary

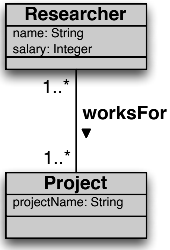
$D_3[Code\colon \text{String}, SSN\colon \text{String}]$
Researchers' Code with SSN

...

Intuitively:

- A researcher should be created from her SSN: **person**(*SSN*)
- A project should be created from its name: **proj**(*PrName*)

Mapping assertions – Example



D_1[*SSN*: String, *PrName*: String]
Researchers and Projects they work for

D_2[*Code*: String, *Salary*: Int]
Researchers' code with salary

D_3[*Code*: String, *SSN*: String]
Researchers' code with SSN

...

m_1:
```
SELECT SSN, PrName
FROM D1
```
$\rightsquigarrow$ Researcher(**person**(*SSN*)),
Project(**proj**(*PrName*)),
projectName(**proj**(*PrName*), *PrName*),
worksFor(**person**(*SSN*), **proj**(*PrName*))

m_2:
```
SELECT SSN, Salary
FROM D2, D3
WHERE D2.Code = D3.Code
```
$\rightsquigarrow$ Researcher(**person**(*SSN*)),
salary(**person**(*SSN*), *Salary*)

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.

[1] `http://d2rq.org/d2rq-language`
[2] `http://sparqlify.org/wiki/Sparqlification_mapping_language`
[3] `https://github.com/ontop/ontop/wiki/ObdalibObdaTurtlesyntax`

Outline

unibz

Integrating complementary information

Common issue in data integration:

- Complementary information about the same entity is distributed over several data sources.
- In different data sources the same entity is represented using different identifiers (URIs).

Problems to address:

- Entity resolution: which data records represent the same entity?
  We do not deal with this aspect here, and assume that information about entity linkage is already available.
- **Integrated querying**: answer queries that require to integrate data about the same entity coming from different data sources.

Approaches to integrated querying

1. Choose a single representation, and physically merge the information into a single data source.

   Requires full control over the data sources.

2. Virtually merge the data, by consistently generating only one URI per real world entity.

   Does not scale well:

   - It requires a central authority for defining URI schemas.
   - For efficiency of OBDI, URIs should be generated from primary keys of the data sources, which typically differ.

3. **Explicitly represent the links between database records resulting from entity resolution.**

Entity linking

Problems to address:

- Links over database identifiers should be represented using OWL `sameAs`.
- `sameAs` is inherently transitive, hence we lose rewritability of queries over the ontology into SQL (i.e., FOL) queries over the sources.
- Also rewritability of consistency checks is lost.
- Performance becomes a critical factor for scalability over large ontologies and Big Data.

Integrating data at Statoil

Databases at Statoil:

- Exploration and Production Data Store (EPDS):
  - Statoil-internal legacy SQL (Oracle 10g) database
  - over 1500 tables (some of them with up to 10 million tuples)
  - 1600 views
  - 700 Gb of data
- NPD FactPages:
  - dataset provided by the Norwegian government
  - contains information on the petroleum activities on the Norwegian continental shelf
- OpenWorks Databases
  - contain projects data produced by geoscientists at Statoil

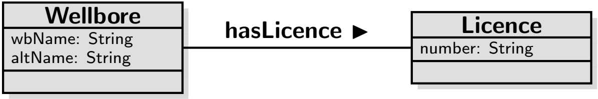
Note:

- *Information in these databases overlap.*
- *They refer to the same entities (companies, wells, licenses) with different identifiers.*

unibz

Example from the oil domain

Ontology:



Data sources:

- D_1, D_2, D_3 contain information about wellbores.
- D_4 contains information about licences.

D_1

id1	name
a1	'A'
a2	'B'
a3	'H'

D_2

id2	name	Well
b1	null	1
b2	'C'	2
b6	'B'	3

D_3

id3	aName
c3	'U1'
c4	'U2'
c5	'U6'

D_4

id4	lNum
9	'Z1'
8	'Z2'
7	'Z3'

Mappings:

- *Wellbore* and *wbName* are defined using D_1 and D_2.
- *altName* is defined using D_3.
- *hasLicense* is defined using D_4.

Moreover, URIs for wellbores from source D_k are generated as **wb**$k(id)$.

Example – Linking information

D_1

id1	name
a1	'A'
a2	'B'
a3	'H'

D_2

id2	name	Well
b1	null	1
b2	'C'	2
b6	'B'	3

D_3

id3	aName
c3	'U1'
c4	'U2'
c5	'U6'

D_4

id4	lNum
9	'Z1'
8	'Z2'
7	'Z3'

Wellbores are cross-linked between datasets as follows:

D_1 and D_2

id1	id2
a1	b2
a2	b1

D_2 and D_3

id2	id3
b1	c4
b2	c3

D_1 and D_3

id1	id3
a3	c5

The cross-links are specified in terms of a set $\mathcal{A}_S$ of sameAs statements:

$$\mathtt{sameAs}(\mathbf{wb1}(\mathtt{a1}), \mathbf{wb2}(\mathtt{b2})), \quad \mathtt{sameAs}(\mathbf{wb2}(\mathtt{b1}), \mathbf{wb3}(\mathtt{c4})), \quad \ldots$$

unibz

Example – Query

D_1

id1	name
a1	'A'
a2	'B'
a3	'H'

D_2

id2	name	Well
b1	null	1
b2	'C'	2
b6	'B'	3

D_3

id3	aName
c3	'U1'
c4	'U2'
c5	'U6'

D_4

id4	lNum
9	'Z1'
8	'Z2'
7	'Z3'

with: a1 ∼ b2 ∼ c3 a2 ∼ b1 ∼ c4 a3 ∼ c5

Consider the query: return all the wellbores and their names.

According to the entailment regime for SPARQL queries, the answer should be all the combinations of equivalent wellbore ids and names:

(**wb1**(a1), 'A'), (**wb2**(b2), 'A'), (**wb3**(c3), 'A'),
(**wb1**(a1), 'C'), (**wb2**(b2), 'C'), (**wb3**(c3), 'C'),
(**wb1**(a2), 'B'), (**wb2**(b1), 'B'), (**wb3**(c4), 'B'),
(**wb1**(a3), 'H'), (**wb3**(c5), 'H')

We want the system to return this answer by evaluating a suitable SQL query over the data sources.

A simple solution based on partial materialization

We have to deal with the inherent semantics of sameAs, which is an equivalence relation:

- We replace the set $\mathcal{A}_S$ of sameAs statements with its transitive, symmetric, and reflexive closure $\mathcal{A}_S^*$.
- However, we do not expand also the (virtual) ABox statements.
- Instead, we rewrite each atom of the input query considering sameAs:

$$\begin{array}{rcl} A(v) & \rightsquigarrow & \texttt{sameAs}(v,x), A(x) \\ P(v,w) & \rightsquigarrow & \texttt{sameAs}(v,x), P(x,y), \texttt{sameAs}(y,w) \end{array}$$

where x, y are fresh existentially quantified variables (actually, blank nodes in SPARQL).

Let $rew_s(q)$ be such a sameAs-rewriting of a SPARQL query q.

Properties of the approach

Correctness of the approach

Let q be aSPARQL query and $rew_S(q)$ its `sameAs`-rewriting. Then

$$cert_{DL}(\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}_S \rangle, q) = cert_{QL}(\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}_S^* \rangle, rew_s(q))$$

However, this approach is only theoretical:

- It requires to pre-compute and materialize $\mathcal{A}_S^*$, which might be prohibitive.
- The linking information is usually not given in the form of `sameAs` statements, but is stored in a database, in suitable tables.

unibz

Linking tables

Towards a practical approach, we consider the following setting:

- The data is divided into different datasets $D_1, \ldots, D_n$, where in each dataset entities are uniquely identified.
- The data belongs to different **categories** $C_1, \ldots, C_m$ (e.g., wellbores, companies, ...):
  - a category corresponds to a set of data records that can be mapped to individuals in the ontology that can in principle be joined;
  - the categories are pairwise disjoint.
- The linking information is stored in linking tables:
  - For each category C, there is a database D^C of linking tables for C.
  - A linking table L^C_{ij} in D^C contains the information about the linkage of entities of category C in datasets D_i and D_j.

unibz

Linking tables – Assumptions

We further impose constraints on the structure of the linking tables:

1. All the information about which objects of category C are linked in datasets D_i and D_j is contained in L^C_{ij}.

   Formally: If there are tables L^C_{ij}, L^C_{ik} and L^C_{kj}, then L^C_{ij} contains all the tuples in $\pi_{id_i,id_j}(L^C_{ik} \bowtie L^C_{kj})$, when evaluated over D^C.

   Example:

   D_1 and D_2

id1	id2
a1	b2
a2	b1

   D_2 and D_3

id2	id3
b1	c4
b2	c3

   D_1 and D_3

id1	id3
a3	c5
a1	c3
a2	c4

2. Linking tables cannot state equality between elements in one dataset.

   Formally: For no join $L^C_{ik} \bowtie \cdots \bowtie L^C_{ni}$, we have that (o, o'), with $o \neq o'$, occurs in $\pi_{L^C_{ik}.id_i, L^C_{ni}.id_i}(L^C_{ik} \bowtie \cdots \bowtie L^C_{ni})$, when evaluated over D^C.

   Note: This amounts to making the Unique Name Assumption for the objects retrieved by the mappings from one dataset.

unibz

Dealing with sameAs through mappings

To minimize the impact of sameAs in the rewriting, we generate the sameAs statements through suitable mappings from the linking tables:

- We choose a specific URI template $\mathtt{uri}_{C,D_i}(id_i)$ for each pair category C – dataset D_i.
- To generate (the virtual sameAs ABox) $\mathcal{A}_S$, for each category C and each linking table L_{ij}^C we extend $\mathcal{M}$ with:

  $$\mathtt{sameAs}(\mathtt{uri}_{C,D_i}(id_i),\ \mathtt{uri}_{C,D_j}(id_j)) \longleftarrow \mathtt{SELECT}\ id_i,\ id_j\ \mathtt{FROM}\ L_{ij}^C$$

- To avoid explicitly adding $\mathcal{A}_S^*$, we embed also the axioms for transitivity and symmetry in the mapping. (For transitivity, this can be done with FOL queries due to the assumptions on the linking tables.)
- We avoid to encode reflexivity, since it would negatively affect performance. This can be done by slightly extending the sameAs query rewriting making use of union.

We have implemented the above techniques in the Ontop OBDA/OBDI, and have successfully adopted it to integrate Statoil data.
For details, see [C_ *et al.*, 2015].

unibz

The Ontop OBDA/OBDI 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 5.x
- Provides a SPARQL end-point (via Sesame Workbench)
- Apache open source license

Outline

1. Ontology-based data integration framework
2. Query answering in OBDI
3. Ontology languages for OBDA
4. Mapping the data to the ontology
5. Object identity
6. Conclusions

Conclusions

- Ontology-based data integration 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 extensive work.
- We have seen some of the techniques required to deal with entity linking in real-world OBDI scenarios.
- In general, 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.
- 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.

unibz

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)*, 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.

unibz

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)*, 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)*, 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)*, pages 602–607, 2005.

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)*, 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)*, 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)*, pages 391–396, 2007.

unibz

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)*, 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.

unibz

References VI

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

[C_ *et al.*, 2015] Diego C_, Martin Giese, Dag Hovland, and Martin Rezk.
Ontology-based integration of cross-linked datasets.
In *Proc. of the 14th Int. Semantic Web Conf. (ISWC)*, Lecture Notes in Computer Science. Springer, 2015.
To appear.

[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)*, 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. Conf. on Logics in Artificial Intelligence (JELIA)*, pages 166–179, 2008.

References VII

[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)*, pages 759–764, 2009.

[Glimm *et al.*, 2008a] 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)*, pages 252–262, 2008.

[Glimm *et al.*, 2008b] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler.
Conjunctive query answering for the description logic $\mathcal{SHIQ}$.
J. of Artificial Intelligence Research, 31:151–198, 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)*, pages 254–263, 2012.

[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.

References VIII

[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.

[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)*, 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)*, 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.

References IX

[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.

[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.