End-User Access to Big Data Using Ontologies

Part 3: Ontology Based Data Access

Diego Calvanese

Free University of Bozen-Bolzano

unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

International Winter School on Big Data
Tarragona, Spain, January 26–30 , 2015

Part 3

Ontology Based Data Access

unibz

Outline of Part 3

1. The *DL-Lite* family of tractable Description Logics
   - Basic features of *DL-Lite*
   - Syntax and semantics of *DL-Lite*
   - Members of the *DL-Lite* family
   - Properties of *DL-Lite*
2. Reasoning in *DL-Lite*
   - Query answering over satisfiable ontologies
   - Ontology satisfiability
   - Complexity of reasoning in *DL-Lite*
3. Linking ontologies to relational data
   - The impedance mismatch problem
   - Ontology-Based Data Access systems
   - Query answering in Ontology-Based Data Access systems
   - Optimizing OBDA in Ontop
4. Conclusions and further work

unibz

Outline of Part 3

unibz

Outline of Part 3

1. The *DL-Lite* family of tractable Description Logics
   - Basic features of *DL-Lite*
   - Syntax and semantics of *DL-Lite*
   - Members of the *DL-Lite* family
   - Properties of *DL-Lite*
2. Reasoning in *DL-Lite*
3. Linking ontologies to relational data
4. Conclusions and further work

The *DL-Lite* family

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

The *DL-Lite* family provides new foundations for Ontology-Based Data Access.

Basic features of *DL-Lite*$_{\mathcal{A}}$

DL-Lite$_{\mathcal{A}}$ is an expressive member of the *DL-Lite* family.

- Takes into account the distinction between **objects** and **values**:
  - Objects are elements of an abstract interpretation domain.
  - Values are elements of concrete data types, such as integers, strings, ecc.
  - Values are connected to objects through **attributes** (rather than roles).
- Captures most of UML class diagrams and Extended ER diagrams.
- Enjoys nice computational properties, both w.r.t. the traditional reasoning tasks, and w.r.t. query answering (see later).

The OWL 2 QL Profile

OWL 2 defines three **profiles**: OWL 2 QL, OWL 2 EL, OWL 2 RL [Motik *et al.*, 2009]

- Each profile corresponds to a syntactic fragment (i.e., a sub-language) of OWL 2 DL that is targeted towards a specific use.
- The restrictions in each profile guarantee better computational properties than those of OWL 2 DL.

The **OWL 2 QL** profile is derived from the DLs of the *DL-Lite* family:

- "[It] includes most of the main features of conceptual models such as UML class diagrams and ER diagrams."
- "[It] is aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task. In OWL 2 QL, conjunctive query answering can be implemented using conventional relational database systems."

Outline of Part 3

unibz

Syntax of the *DL-Lite*$_{\mathcal{A}}$ description language

- Role expressions (*object properties* in OWL 2 QL)
  - atomic role: P
  - basic role: $Q \ ::= \ P \mid P^-$
  - arbitrary role: $R \ ::= \ Q \mid \neg Q$ (to express disjointness)
- Concept expressions (*classes* in OWL 2 QL):
  - atomic concept: A
  - basic concept: $B \ ::= \ A \mid \exists Q \mid \delta(U)$
  - arbitrary concept: $C \ ::= \ \top_C \mid B \mid \neg B$ (to express disjointness)
- Attribute expressions (*data properties* in OWL 2 QL):
  - atomic attribute: U
  - arbitrary attribute: $V \ : = \ U \mid \neg U$ (to express disjointness)
- Value-domain expressions (*datatypes* in OWL 2 QL):
  - attribute range: $\rho(U)$
  - RDF datatypes: T_i
  - top domain: $\top_D$

unibz

Semantics of *DL-Lite*$_{\mathcal{A}}$ – Objects vs. values

	Objects	Values
Interpretation domain $\Delta^{\mathcal{I}}$	Domain of objects $\Delta_O^{\mathcal{I}}$	Domain of values $\Delta_V^{\mathcal{I}}$
Alphabet Γ of constants	Object constants Γ_O	Value constants Γ_V
	$c^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$	$d^{\mathcal{I}} = \mathit{val}(d)$ given a priori
Unary predicates	Concept C	RDF datatype T_i
	$C^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}}$	$T_i^{\mathcal{I}} \subseteq \Delta_V^{\mathcal{I}}$ given a priori
Binary predicates	Role R	Attribute V
	$R^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}$	$V^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$

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

Construct	Syntax	Example	Semantics
atomic role	P	child	$P^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}$
inverse role	P^-	child$^-$	$\{(o,o') \mid (o',o) \in P^{\mathcal{I}}\}$
role negation	$\neg Q$	$\neg$manages	$(\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$
atomic concept	A	Doctor	$A^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}}$
existential restriction	$\exists Q$	$\exists$child$^-$	$\{o \mid \exists o'. (o,o') \in Q^{\mathcal{I}}\}$
concept negation	$\neg B$	$\neg\exists$child	$\Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$
attribute domain	$\delta(U)$	δ(salary)	$\{o \mid \exists v. (o,v) \in U^{\mathcal{I}}\}$
top concept	$\top_C$		$\top_C^{\mathcal{I}} = \Delta_O^{\mathcal{I}}$
atomic attribute	U	salary	$U^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$
attribute negation	$\neg U$	$\neg$salary	$(\Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus U^{\mathcal{I}}$
top domain	$\top_D$		$\top_D^{\mathcal{I}} = \Delta_V^{\mathcal{I}}$
datatype	T_i	`xsd:int`	$T_i^{\mathcal{I}} \subseteq \Delta_V^{\mathcal{I}}$ (predefined)
attribute range	$\rho(U)$	ρ(salary)	$\{v \mid \exists o. (o,v) \in U^{\mathcal{I}}\}$
object constant	c	`john`	$c^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$
value constant	d	`'john'`	*val*$(d) \in \Delta_V^{\mathcal{I}}$ (predefined)

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

TBox assertions can have the following forms:

- Inclusion assertions (also called positive inclusions):

$B_1 \sqsubseteq B_2$	concept inclusion	$\rho(U) \sqsubseteq T_i$	value-domain inclusion
$Q_1 \sqsubseteq Q_2$	role inclusion	$U_1 \sqsubseteq U_2$	attribute inclusion

- Disjointness assertions (also called negative inclusions):

$B_1 \sqsubseteq \neg B_2$	concept disjointness		
$Q_1 \sqsubseteq \neg Q_2$	role disjointness	$U_1 \sqsubseteq \neg U_2$	attribute disjointness

- Functionality assertions:

$(\textbf{funct}\ Q)$	role functionality	$(\textbf{funct}\ U)$	attribute functionality

- Identification assertions: $(\textbf{id}\ B\ I_1, \ldots, I_n)$
  where each I_j is a role, an inverse role, or an attribute

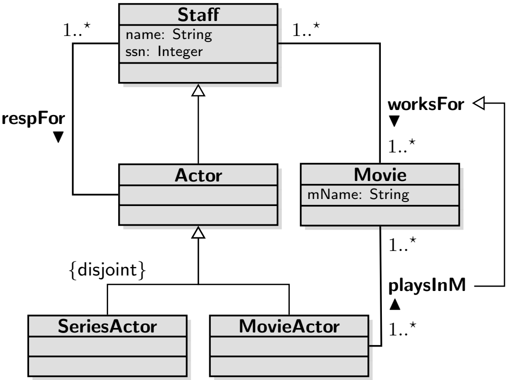
ABox assertions: $A(c)$, $P(c, c')$, $U(c, d)$,
where c, c' are object constants and d is a value constant

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

Assertion	Syntax	Example	Semantics
conc. incl.	$B_1 \sqsubseteq B_2$	Father $\sqsubseteq$ $\exists$child	$B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}$
role incl.	$Q_1 \sqsubseteq Q_2$	father $\sqsubseteq$ anc	$Q_1^{\mathcal{I}} \subseteq Q_2^{\mathcal{I}}$
v.dom. incl.	$\rho(U) \sqsubseteq T_i$	ρ(age) $\sqsubseteq$ `xsd:int`	$\rho(U)^{\mathcal{I}} \subseteq T_i^{\mathcal{I}}$
attr. incl.	$U_1 \sqsubseteq U_2$	offPhone $\sqsubseteq$ phone	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
conc. disj.	$B_1 \sqsubseteq \neg B_2$	Person $\sqsubseteq$ $\neg$Course	$B_1^{\mathcal{I}} \subseteq (\neg B_2)^{\mathcal{I}}$
role disj.	$Q_1 \sqsubseteq \neg Q_2$	sibling $\sqsubseteq$ $\neg$cousin	$Q_1^{\mathcal{I}} \subseteq (\neg Q_2)^{\mathcal{I}}$
attr. disj.	$U_1 \sqsubseteq \neg U_2$	offPhn $\sqsubseteq$ $\neg$homePhn	$U_1^{\mathcal{I}} \subseteq (\neg U_2)^{\mathcal{I}}$
role funct.	(**funct** Q)	(**funct** father)	$\forall o, o_1, o_2.(o, o_1) \in Q^{\mathcal{I}} \wedge (o, o_2) \in Q^{\mathcal{I}} \rightarrow o_1 = o_2$
att. funct.	(**funct** U)	(**funct** ssn)	$\forall o, v, v'.(o, v) \in U^{\mathcal{I}} \wedge (o, v') \in U^{\mathcal{I}} \rightarrow v = v'$
id const.	(**id** B $I_1, \ldots, I_n$)	(**id** Person name, dob)	$I_1, \ldots, I_n$ identify instances of B
mem. asser.	$A(c)$	Father(bob)	$c^{\mathcal{I}} \in A^{\mathcal{I}}$
mem. asser.	$P(c_1, c_2)$	child(bob, ann)	$(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in P^{\mathcal{I}}$
mem. asser.	$U(c, d)$	phone(bob, '2345')	$(c^{\mathcal{I}}, \mathit{val}(d)) \in U^{\mathcal{I}}$

unibz

DL-Lite$_{\mathcal{A}}$ – Example



$$
\begin{array}{rcl}
\text{Actor} & \sqsubseteq & \text{Staff} \\
\text{SeriesActor} & \sqsubseteq & \text{Actor} \\
\text{MovieActor} & \sqsubseteq & \text{Actor} \\
\text{SeriesActor} & \sqsubseteq & \neg\text{MovieActor} \\
\text{Staff} & \sqsubseteq & \delta(\text{ssn}) \\
\delta(\text{ssn}) & \sqsubseteq & \text{Staff} \\
\rho(\text{ssn}) & \sqsubseteq & \texttt{xsd:int} \\
& (\textbf{funct}\ \text{ssn}) & \\
& (\textbf{id}\ \text{Staff ssn}) & \\
\exists\text{worksFor} & \sqsubseteq & \text{Staff} \\
\exists\text{worksFor}^- & \sqsubseteq & \text{Movie} \\
\text{Staff} & \sqsubseteq & \exists\text{worksFor} \\
\text{Movie} & \sqsubseteq & \exists\text{worksFor}^- \\
& (\textbf{funct}\ \text{playsInM}) & \\
& (\textbf{funct}\ \text{playsInM}^-) & \\
\text{playsInM} & \sqsubseteq & \text{worksFor} \\
& \vdots &
\end{array}
$$

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

unibz

Outline of Part 3

unibz

Restriction on TBox assertions in $DL\text{-}Lite_{\mathcal{A}}$ ontologies

We will see that, to ensure the good computational properties that we aim at, we have to impose a **restriction** on the use of functionality and role/attribute inclusions.

Restriction on $DL\text{-}Lite_{\mathcal{A}}$ TBoxes

No functional or identifying role or attribute can be specialized by using it in the right-hand side of a role or attribute inclusion assertion.

Formally:

- If $(\textbf{funct}\ P)$, $(\textbf{funct}\ P^-)$, $(\textbf{id}\ B\ \ldots, P, \ldots)$, or $(\textbf{id}\ B\ \ldots, P^-, \ldots)$ is in $\mathcal{T}$, then $Q \sqsubseteq P$ and $Q \sqsubseteq P^-$ are **not in** $\mathcal{T}$.
- If $(\textbf{funct}\ U)$ or $(\textbf{id}\ B\ \ldots, U, \ldots)$ is in $\mathcal{T}$, then $U' \sqsubseteq U$ is **not in** $\mathcal{T}$.

unibz

DL-Lite$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$

We consider also two sub-languages of *DL-Lite*$_{\mathcal{A}}$ (that trivially obey the previous restriction):

- *DL-Lite*$_{\mathcal{F}}$: Allows for functionality assertions, but does not allow for role inclusion assertions.
- *DL-Lite*$_{\mathcal{R}}$: Allows for role inclusion assertions, but does not allow for functionality assertions.
  This is the DL that corresponds to the OWL 2 QL profile.

Note: We simply use *DL-Lite* to refer to any of the logics of the *DL-Lite* family.

Outline of Part 3

unibz

Capturing basic ontology constructs in $DL\text{-}Lite_{\mathcal{A}}$

ISA between classes	$A_1 \sqsubseteq A_2$
Disjointness between classes	$A_1 \sqsubseteq \neg A_2$
Mandatory participation to relations	$A_1 \sqsubseteq \exists P \quad A_2 \sqsubseteq \exists P^-$
Domain and range of relations	$\exists P \sqsubseteq A_1 \quad \exists P^- \sqsubseteq A_2$
Functionality of relations	$(\textbf{funct } P) \quad (\textbf{funct } P^-)$
ISA between relations	$Q_1 \sqsubseteq Q_2$
Disjointness between relations	$Q_1 \sqsubseteq \neg Q_2$
Domain and range of attributes	$\delta(U) \sqsubseteq A \quad \rho(U) \sqsubseteq T_i$
Mandatory and functional attributes	$A \sqsubseteq \delta(U) \quad (\textbf{funct } U)$
Identification constraints	$(\textbf{id } A\ P, \ldots, P'^-, \ldots, U, \ldots)$

unibz

Properties of *DL-Lite*

- The TBox may contain **cyclic dependencies** (which typically increase the computational complexity of reasoning).

  Example: $A \sqsubseteq \exists P, \quad \exists P^- \sqsubseteq A$

- In the syntax, we have not included $\sqcap$ on the right hand-side of inclusion assertions, but it can trivially be added, since

  $B \sqsubseteq C_1 \sqcap C_2$ is equivalent to $\begin{array}{lcl} B & \sqsubseteq & C_1 \\ B & \sqsubseteq & C_2 \end{array}$

- A domain assertion on role P has the form: $\exists P \sqsubseteq A_1$
  A range assertion on role P has the form: $\exists P^- \sqsubseteq A_2$

unibz

Finite model property

DL-Lite$_\mathcal{F}$ (and *DL-Lite*$_\mathcal{A}$) does **not** enjoy the **finite model property**.

Example

TBox $\mathcal{T}$: $\mathsf{Nat} \sqsubseteq \exists\mathsf{succ}$ $\quad\quad \exists\mathsf{succ}^- \sqsubseteq \mathsf{Nat}$

$\mathsf{Zero} \sqsubseteq \mathsf{Nat} \sqcap \neg\exists\mathsf{succ}^-$ $\quad\quad (\textbf{funct}\ \mathsf{succ}^-)$

ABox $\mathcal{A}$: $\mathsf{Zero}(0)$

$\mathcal{O} = \langle\mathcal{T}, \mathcal{A}\rangle$ admits only infinite models.
Hence, it is satisfiable, but **not finitely satisfiable**.

Hence, reasoning w.r.t. arbitrary models is different from reasoning w.r.t. finite models only.

Notice that instead *DL-Lite*$_\mathcal{R}$ does enjoy the finite model property.

unibz

Outline of Part 3

unibz

Remarks

In the following, we ignore the distinction between objects and values, since it is not relevant for reasoning. Hence we do not use value domains and attributes.

Notation:

- When the distinction between *DL-Lite*$_{\mathcal{R}}$, *DL-Lite*$_{\mathcal{F}}$, or *DL-Lite*$_{\mathcal{A}}$ is not important, we use just *DL-Lite*.
- Q denotes a basic role, i.e., $Q \longrightarrow P \mid P^-$.
- R denotes a general role, i.e., $R \longrightarrow Q \mid \neg Q$.
- C denotes a general concept, i.e., $C \longrightarrow A \mid \neg A \mid \exists Q \mid \neg \exists Q$, where A is an atomic concept.

unibz

Reasoning services

One can show that in *DL-Lite* all TBox reasoning services can be reduced to **ontology satisfiability**.

Hence, in the following, we concentrate on:

- **Ontology satisfiability:** Verify whether an ontology $\mathcal{O}$ is satisfiable, i.e., whether $\mathcal{O}$ admits at least one model.
- **Query answering:** Given a query q over an ontology $\mathcal{O}$, find all tuples $\vec{c}$ of constants such that $\mathcal{O} \models q(\vec{c})$.

Query answering vs. ontology satisfiability

- In the case in which an ontology is unsatisfiable, according to the “ex falso quod libet” principle, reasoning is trivialized.
- In particular, query answering is meaningless, since every tuple is in the answer to every query.
- We are not interested in encoding meaningless query answering into the perfect reformulation of the input query. Therefore, before query answering, we will always check ontology satisfiability to single out meaningful cases.

Thus, we proceed as follows:

1. We show how to do **query answering over satisfiable ontologies**.
2. We show how we can exploit the query answering algorithm also to check ontology satisfiability.

Outline of Part 3

Query answering over satisfiable ontologies

Given a CQ q and a satisfiable ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, we compute $cert(q, \mathcal{O})$ as follows:

1. Using $\mathcal{T}$, **rewrite** q into a UCQ $r_{q,\mathcal{T}}$ (the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. **Evaluate** $r_{q,\mathcal{T}}$ over $\mathcal{A}$ (simply viewed as data), to return $cert(q, \mathcal{O})$.

Correctness of this procedure shows **FOL-rewritability** of query answering in *DL-Lite*.

Positive vs. negative inclusions

We call positive inclusions (PIs) assertions of the form

$$A_1 \sqsubseteq A_2 \qquad \exists Q_1 \sqsubseteq A_2 \qquad Q_1 \sqsubseteq Q_2$$
$$A_1 \sqsubseteq \exists Q_2 \qquad \exists Q_1 \sqsubseteq \exists Q_2$$

We call negative inclusions (NIs) assertions of the form

$$A_1 \sqsubseteq \neg A_2 \qquad \exists Q_1 \sqsubseteq \neg A_2 \qquad Q_1 \sqsubseteq \neg Q_2$$
$$A_1 \sqsubseteq \neg \exists Q_2 \qquad \exists Q_1 \sqsubseteq \neg \exists Q_2$$

unibz

Query rewriting

Consider the query $\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$

Intuition: Use the PIs as basic rewriting rules:

$$\mathsf{AssistantProf} \sqsubseteq \mathsf{Professor}$$

as a logic rule: $\mathsf{Professor}(z) \leftarrow \mathsf{AssistantProf}(z)$

Basic rewriting step:

when an atom in the query unifies with the **head** of the rule,
substitute the atom with the **body** of the rule.

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

In the example, the PI $\mathsf{AssistantProf} \sqsubseteq \mathsf{Professor}$ applies to the atom $\mathsf{Professor}(x)$. Towards the computation of the perfect rewriting, we add to the input query above, the query

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

unibz

Query rewriting (cont'd)

Consider the query $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x,y), \mathsf{Course}(y)$

and the PI $\exists\mathsf{teaches}^- \sqsubseteq \mathsf{Course}$

as a logic rule: $\mathsf{Course}(z_2) \leftarrow \mathsf{teaches}(z_1,z_2)$

The PI applies to the atom $\mathsf{Course}(y)$, and we add to the perfect rewriting the query

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x,y), \mathsf{teaches}(z_1,y)$$

Consider now the query $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x,y)$

and the PI $\mathsf{Professor} \sqsubseteq \exists\mathsf{teaches}$

as a logic rule: $\mathsf{teaches}(z,f(z)) \leftarrow \mathsf{Professor}(z)$

The PI applies to the atom $\mathsf{teaches}(x,y)$, and we add to the perfect rewriting the query

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

unibz

Query rewriting – Constants

Conversely, for the query $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, \mathtt{fl})$

and the same PI as before $\mathsf{Professor} \sqsubseteq \exists\mathsf{teaches}$

as a logic rule: $\mathsf{teaches}(z, f(z)) \leftarrow \mathsf{Professor}(z)$

$\mathsf{teaches}(x, \mathtt{fl})$ does not unify with $\mathsf{teaches}(z, f(z))$, since the **skolem term** $f(z)$ in the head of the rule **does not unify** with the constant $\mathtt{fl}$.
Remember: We adopt the **unique name assumption**.

In this case, we say that the PI does not apply to the atom $\mathsf{teaches}(x, \mathtt{fl})$.

The same holds for the following query, where y is **distinguished**, since unifying $f(z)$ with y would correspond to returning a skolem term as answer to the query:

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

Query rewriting – Join variables

An analogous behavior to the one with constants and with distinguished variables holds when the atom contains **join variables** that would have to be unified with skolem terms.

Consider the query $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$

and the PI $\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$

as a logic rule: $\mathsf{teaches}(z, f(z)) \leftarrow \mathsf{Professor}(z)$

The PI above does **not** apply to the atom $\mathsf{teaches}(x, y)$.

unibz

Query rewriting – Reduce step

Consider now the query $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{teaches}(z, y)$

and the PI $\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$

as a logic rule: $\mathsf{teaches}(z, f(z)) \leftarrow \mathsf{Professor}(z)$

This PI does not apply to $\mathsf{teaches}(x, y)$ or $\mathsf{teaches}(z, y)$, since y is in join, and we would again introduce the skolem term in the rewritten query.

However, we can transform the above query by unifying the atoms $\mathsf{teaches}(x, y)$ and $\mathsf{teaches}(z, y)$. This rewriting step is called **reduce**, and produces the query

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

Now, we can apply the PI above, and add to the rewriting the query

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

Query rewriting – Summary

Reformulate the CQ q into a set of queries:

- Apply to q and the computed queries in all possible ways the PIs in $\mathcal{T}$:

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

  ('_' denotes an **unbound** variable, i.e., a variable that appears only once)

  This corresponds to exploiting ISAs, role typing, and mandatory participation to obtain new queries that could contribute to the answer.

- Apply in all possible ways unification between atoms in a query.

  Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method.

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

unibz

Query rewriting algorithm

Algorithm *PerfectRef*$(Q, \mathcal{T}_P)$
Input: union of conjunctive queries Q, set of *DL-Lite*$_{\mathcal{A}}$ PIs $\mathcal{T}_P$
Output: union of conjunctive queries PR
$PR := Q$;
repeat
  $PR' := PR$;
  for each $q \in PR'$ **do**
    for each g in q **do**
      for each PI I in $\mathcal{T}_P$ **do**
        if I is applicable to g **then** $PR := PR \cup \{$ *ApplyPI*$(q, g, I) \}$;
    for each g_1, g_2 in q **do**
      if g_1 and g_2 unify **then** $PR := PR \cup \{\tau($*Reduce*$(q, g_1, g_2))\}$;
until $PR' = PR$;
return PR

Observations:

- Termination follows from having only finitely many different rewritings.
- NIs or functionalities do not play any role in the rewriting of the query.

Query answering in *DL-Lite* – Example

TBox: $\mathsf{Professor} \sqsubseteq \exists\mathsf{teaches}$
$\exists\mathsf{teaches}^- \sqsubseteq \mathsf{Course}$

Query: $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$

Perfect Rewriting: $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{teaches}(\_, y)$
$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, \_)$
$\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$

ABox: $\mathsf{teaches}(\mathtt{john}, \mathtt{fl})$
$\mathsf{Professor}(\mathtt{mary})$

It is easy to see that evaluating the perfect rewriting over the ABox viewed as a database produces as answer $\{\mathtt{john}, \mathtt{mary}\}$.

unibz

Query answering in *DL-Lite* – An interesting example

TBox: $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather}$
$\exists \mathsf{hasFather}^- \sqsubseteq \mathsf{Person}$

ABox: $\mathsf{Person}(\mathtt{mary})$

Query: $\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, y_3)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, \_)$

$\Downarrow$ **Apply** $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather}$ to the atom $\mathsf{hasFather}(y_2, \_)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{Person}(y_2)$

$\Downarrow$ **Apply** $\exists \mathsf{hasFather}^- \sqsubseteq \mathsf{Person}$ to the atom $\mathsf{Person}(y_2)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(\_, y_2)$

$\Downarrow$ **Unify** atoms $\mathsf{hasFather}(y_1, y_2)$ and $\mathsf{hasFather}(\_, y_2)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2)$

$\Downarrow$

...

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

$\Downarrow$ **Apply** $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather}$ to the atom $\mathsf{hasFather}(x, \_)$

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

Query answering over satisfiable *DL-Lite* ontologies

For an ABox $\mathcal{A}$ and a query q over $\mathcal{A}$, let $\mathit{Eval}_{\text{CWA}}(q, \mathcal{A})$ denote the evaluation of q over $\mathcal{A}$ considered as a database (i.e., considered under the CWA).

Theorem

Let $\mathcal{T}$ be a *DL-Lite* TBox, $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$, and q a CQ over $\mathcal{T}$. Then, for each ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, we have that

$$cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathit{Eval}_{\text{CWA}}(\mathit{PerfectRef}(q, \mathcal{T}_P), \mathcal{A}).$$

As a consequence, query answering over a satisfiable *DL-Lite* ontology is FOL-rewritable.

Notice that we did not use NIs or functionality assertions of $\mathcal{T}$ in computing $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle$. Indeed, **when the ontology is satisfiable, we can ignore NIs and functionality assertions for query answering**.

Canonical model of a *DL-Lite* ontology

The proof of the previous result exploits a fundamental property of *DL-Lite*, that relies on the following notion.

Def.: Canonical model

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite* ontology. A model $\mathcal{I}_{\mathcal{O}}$ of $\mathcal{O}$ is called **canonical** if for every model $\mathcal{I}$ of $\mathcal{O}$ there is a homomorphism from $\mathcal{I}_{\mathcal{O}}$ to $\mathcal{I}$.

Theorem

Every satisfiable *DL-Lite* ontology has a **canonical model**.

Properties of the canonical models of a *DL-Lite* ontology:

- A canonical model is in general infinite.
- All canonical models are homomorphically equivalent, hence we can do as if there was a single canonical model.

Query answering in *DL-Lite* – Canonical model

From the definition of canonical model, and since homomorphisms are closed under composition, we get that:

To compute the certain answer to a query q over an ontology $\mathcal{O}$, one could in principle evaluate q over a canonical model $\mathcal{I}_{\mathcal{O}}$ of $\mathcal{O}$.

- This does not give us directly an algorithm for query answering over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, since $\mathcal{I}_{\mathcal{O}}$ may be infinite.
- However, one can show that evaluating q over $\mathcal{I}_{\mathcal{O}}$ amounts to evaluating the perfect rewriting $r_{q,\mathcal{T}}$ over $\mathcal{A}$.

unibz

Outline of Part 3

Satisfiability of ontologies with only PIs

Let us now consider the problem of establishing whether an ontology is satisfiable.

A first notable result tells us that PIs alone cannot generate ontology unsatisfiability.

Theorem

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite* ontology where $\mathcal{T}$ contains **only PIs**.
Then, $\mathcal{O}$ is satisfiable.

Satisfiability of $DL\text{-}Lite_{\mathcal{A}}$ ontologies

Unsatisfiability in $DL\text{-}Lite_{\mathcal{A}}$ ontologies can be caused by **NIs** or by **functionality assertions**.

Example

TBox $\mathcal{T}$:
$\mathsf{Professor} \sqsubseteq \neg\mathsf{Student}$
$\exists\mathsf{teaches} \sqsubseteq \mathsf{Professor}$
$(\textbf{funct}\ \mathsf{teaches}^-)$

ABox $\mathcal{A}$:
$\mathsf{Student}(\mathtt{john})$
$\mathsf{teaches}(\mathtt{john}, \mathtt{fl})$
$\mathsf{teaches}(\mathtt{michael}, \mathtt{fl})$

unibz

Checking satisfiability of *DL-Lite*$_{\mathcal{A}}$ ontologies

Satisfiability of a *DL-Lite*$_{\mathcal{A}}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is reduced to evaluating over *DB*$(\mathcal{A})$ a UCQ that asks for the **existence of objects violating the NI and functionality assertions**.

Let $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$.
We deal with NIs and functionality assertions differently.

For each NI $N \in \mathcal{T}$:

1. we construct a boolean CQ $q_N()$ such that

   $$\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N() \quad \text{iff} \quad \langle \mathcal{T}_P \cup \{N\}, \mathcal{A} \rangle \text{ is unsatisfiable}$$

2. We check whether $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$ using *PerfectRef*, i.e., we compute *PerfectRef*$(q_N, \mathcal{T}_P)$, and evaluate it over *DB*$(\mathcal{A})$.

For each functionality assertion $F \in \mathcal{T}$:

1. we construct a boolean CQ $q_F()$ such that

   $$\mathcal{A} \models q_F() \quad \text{iff} \quad \langle \{F\}, \mathcal{A} \rangle \text{ is unsatisfiable.}$$

2. We check whether $\mathcal{A} \models q_F()$, by simply evaluating q_F over *DB*$(\mathcal{A})$.

Checking violations of negative inclusions

For each **NI** N in $\mathcal{T}$ we compute a boolean CQ $q_N()$ according to the following rules:

$$
\begin{array}{rcl}
A_1 \sqsubseteq \neg A_2 & \rightsquigarrow & q_N() \leftarrow A_1(x), A_2(x) \\
\exists P \sqsubseteq \neg A \ \text{ or } \ A \sqsubseteq \neg \exists P & \rightsquigarrow & q_N() \leftarrow P(x,y), A(x) \\
\exists P^- \sqsubseteq \neg A \ \text{ or } \ A \sqsubseteq \neg \exists P^- & \rightsquigarrow & q_N() \leftarrow P(y,x), A(x) \\
\exists P_1 \sqsubseteq \neg \exists P_2 & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(x,z) \\
\exists P_1 \sqsubseteq \neg \exists P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(z,x) \\
\exists P_1^- \sqsubseteq \neg \exists P_2 & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(y,z) \\
\exists P_1^- \sqsubseteq \neg \exists P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(z,y) \\
P_1 \sqsubseteq \neg P_2 \ \text{ or } \ P_1^- \sqsubseteq \neg P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(x,y) \\
P_1^- \sqsubseteq \neg P_2 \ \text{ or } \ P_1 \sqsubseteq \neg P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(y,x)
\end{array}
$$

unibz

Checking violations of negative inclusions – Example

PIs $\mathcal{T}_P$: $\exists$teaches $\sqsubseteq$ Professor
NIs N : Professor $\sqsubseteq$ $\neg$Student

Query q_N: $q_N() \leftarrow \text{Student}(x), \text{Professor}(x)$

Perfect Rewriting: $q_N() \leftarrow \text{Student}(x), \text{Professor}(x)$
$q_N() \leftarrow \text{Student}(x), \text{teaches}(x, \_)$

ABox $\mathcal{A}$: $\text{teaches}(\texttt{john}, \texttt{fl})$
$\text{Student}(\texttt{john})$

It is easy to see that $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$, and that the ontology $\langle \mathcal{T}_P \cup \{\text{Professor} \sqsubseteq \neg\text{Student}\},\ \mathcal{A} \rangle$ **is unsatisfiable**.

Boolean queries vs. non-boolean queries for NIs

To ensure correctness of the method, the queries used to check for the violation of a NI need to be **boolean**.

Example

TBox $\mathcal{T}$: $A_1 \sqsubseteq \neg A_0$ $\quad \exists P \sqsubseteq A_1$
$A_1 \sqsubseteq A_0$ $\quad A_2 \sqsubseteq \exists P^-$

ABox $\mathcal{A}$: $A_2(c)$

Since A_1, P, and A_2 are unsatisfiable, also $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable.

Consider the query corresponding to the NI $A_1 \sqsubseteq \neg A_0$.

$q_N() \leftarrow A_1(x), A_0(x)$

Then *PerfectRef*($q_N, \mathcal{T}_P$) is:

$q_N() \leftarrow A_1(x), A_0(x)$
$q_N() \leftarrow A_1(x)$
$q_N() \leftarrow P(x, \_)$
$q_N() \leftarrow A_2(\_)$

We have that $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$.

$q'_N(x) \leftarrow A_1(x), A_0(x)$

Then *PerfectRef*($q'_N, \mathcal{T}_P$) is

$q'_N(x) \leftarrow A_1(x), A_0(x)$
$q'_N(x) \leftarrow A_1(x)$
$q'_N(x) \leftarrow P(x, \_)$

$cert(q'_N, \langle \mathcal{T}_P, \mathcal{A} \rangle) = \emptyset$, hence $q'_N(x)$ does not detect unsatisfiability.

Checking violations of functionality assertions

For each **functionality assertion** F in $\mathcal{T}$ we compute a boolean FOL query $q_F()$ according to the following rules:

$$
\begin{array}{lcl}
(\textbf{funct } P) & \rightsquigarrow & q_F() \leftarrow P(x,y), P(x,z), y \neq z \\
(\textbf{funct } P^-) & \rightsquigarrow & q_F() \leftarrow P(x,y), P(z,y), x \neq z
\end{array}
$$

Example

Functionality F: $(\textbf{funct teaches}^-)$

Query q_F: $q_F() \leftarrow \textbf{teaches}(x,y), \textbf{teaches}(z,y), x \neq z$

ABox $\mathcal{A}$: $\text{teaches}(\texttt{john}, \texttt{fl})$
$\text{teaches}(\texttt{michael}, \texttt{fl})$

It is easy to see that $\mathcal{A} \models q_F()$, and that $\langle\{(\textbf{funct teaches}^-)\}, \mathcal{A}\rangle$, **is unsatisfiable**.

unibz

From satisfiability to query answering in $\textit{DL-Lite}_{\mathcal{A}}$

Lemma (Separation for $\textit{DL-Lite}_{\mathcal{A}}$)

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\textit{DL-Lite}_{\mathcal{A}}$ ontology, and $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$. Then, $\mathcal{O}$ is unsatisfiable iff one of the following condition holds:

- (a) There exists a NI $N \in \mathcal{T}$ such that $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$.
- (b) There exists a functionality assertion $F \in \mathcal{T}$ such that $\mathcal{A} \models q_F()$.

(a) relies on the properties that **NIs do not interact with each other**, and that **interaction between NIs and PIs** is captured **through** *PerfectRef*.

(b) exploits the property that **NIs and PIs do not interact with functionalities**: indeed, no functionality assertion is contradicted in a $\textit{DL-Lite}_{\mathcal{A}}$ ontology $\mathcal{O}$, beyond those explicitly contradicted by the ABox.

Notably, to check ontology satisfiability, each NI and each functionality assertion can be processed individually.

FOL-rewritability of satisfiability in $DL\text{-}Lite_{\mathcal{A}}$

From the previous lemma and the theorem on query answering for satisfiable $DL\text{-}Lite_{\mathcal{A}}$ ontologies, we get the following result.

Theorem

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $DL\text{-}Lite_{\mathcal{A}}$ ontology, and $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$.
Then, $\mathcal{O}$ is unsatisfiable iff one of the following condition holds:

(a) There exists a NI $N \in \mathcal{T}$ s.t. $\mathit{Eval}_{\text{CWA}}(\mathit{PerfectRef}(q_N, \mathcal{T}_P), \mathcal{A})$ returns *true*.

(b) There exists a func. assertion $F \in \mathcal{T}$ s.t. $\mathit{Eval}_{\text{CWA}}(q_F, \mathcal{A})$ returns *true*.

Note: All the queries $q_N()$ and $q_F()$ can be combined into a single UCQ. Hence, satisfiability of a $DL\text{-}Lite_{\mathcal{A}}$ ontology is reduced to evaluating a FOL-query over an ontology whose TBox consists of positive inclusions only (and hence is satisfiable).

unibz

Outline of Part 3

Complexity results for *DL-Lite*

TBox reasoning

- PTime in the size of the **TBox** (schema complexity)

Ontology satisfiability

- PTime in the size of the **ontology** (combined complexity)
- AC^0 in the size of the **ABox** (data complexity)

Query answering

- NP-**complete** in the size of query and ontology (combined complexity)
- PTime in the size of the **ontology** (schema+data complexity)
- AC^0 in the size of the **ABox** (data complexity)

Can we further extend these results to more expressive ontology languages?
Essentially NO!
(unless we take particular care)

Beyond $DL\text{-}Lite_{\mathcal{A}}$: results on data complexity

Essentially all extensions of *DL-Lite* with additional DL constructs, or with combinations of constructs that are not legal in *DL-Lite*, make it lose its nice computational properties [Calvanese *et al.*, 2013b].

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

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

unibz

Combining functionalities and role inclusions

One can show that, by including in *DL-Lite* both functionality of roles and role inclusions **without restrictions on their interaction** [Artale *et al.*, 2009]:

- query answering becomes PTime-hard in data complexity;
- the complexity of TBox reasoning jumps from NLogSpace to ExpTime-complete.

Recall that, when the data complexity of query answering is above AC^0, the DL does not enjoy FOL-rewritability.

As a consequence of these results, we get:

The restriction on the interaction of functionality and role inclusions of *DL-Lite*$_\mathcal{A}$ is necessary:

- to preserve FOL-rewritability of query answering and ontology satisfiability;
- to guarantee efficient reasoning on the TBox (i.e., at the schema level).

unibz

Outline of Part 3

Outline of Part 3

Data in external sources

There are several situations where the assumptions of having the data in an ABox managed directly by the ontology system (e.g., a Description Logics reasoner) is not feasible or realistic:

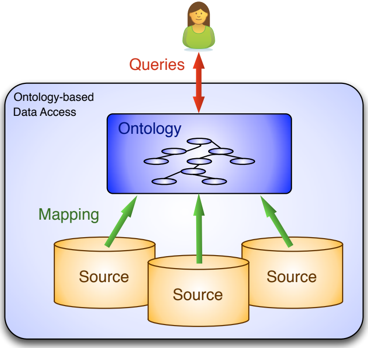
- When the ABox is very large, so that it requires relational database technology.
- When we have no direct control over the data since it belongs to some external organization, which controls the access to it.
- When multiple data sources need to be accessed, such as in Information Integration.

We would like to deal with such a situation by keeping the data in the external (relational) storage, and performing **query answering** by leveraging the capabilities of the **relational engine**.

Ontology-based data access: Architecture

The architecture of an OBDA system 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.



unibz

The impedance mismatch problem

We have to deal with the **impedance mismatch problem**:

- Sources store data, which is constituted by values taken from concrete domains, such as strings, integers, codes, ...
- Instead, instances of concepts and relations in an ontology are (abstract) objects.

Solution:

- We need to specify how to construct from the data values in the relational sources the (abstract) objects that populate the ABox of the ontology.
- This specification is embedded in the mappings between the data sources and the ontology.

Note: the **ABox** is only **virtual**, and the objects are not materialized.

unibz

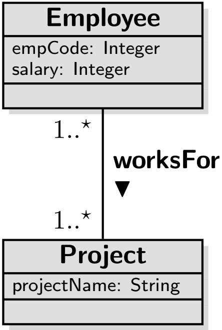
Solution to the impedance mismatch problem

We need to define a **mapping language** that allows for specifying how to transform data into abstract objects:

- Each mapping assertion maps:
  - a query that retrieves values from a data source to ...
  - a set of atoms specified over the ontology.
- Basic idea: use **Skolem functions** in the atoms over the ontology to "generate" the objects from the data values.
- Semantics of mappings:
  - Objects are denoted by terms (of exactly one level of nesting).
  - Different terms denote different objects (i.e., we make the unique name assumption on terms).

unibz

Impedance mismatch – Example



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

D_1[*SSN*: String, *PrName*: String]
Employees and projects they work for

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

D_3[*Code*: String, *SSN*: String]
Employee's Code with SSN

...

Intuitively:

- An employee should be created from her SSN: **pers**(*SSN*)
- A project should be created from its name: **proj**(*PrName*)

Creating object identifiers

We need to associate to the data in the tables objects in the ontology.

- We introduce an alphabet Λ of **function symbols**, each with an associated arity.
- To denote values, we use value constants from an alphabet Γ_V.
- To denote objects, we use **object terms** instead of object constants.
  An object term has the form $\mathbf{f}(d_1, \ldots, d_n)$, with $\mathbf{f} \in \Lambda$, and each d_i a value constant in Γ_V.

Example

- If a person is identified by her *SSN*, we can introduce a function symbol $\mathbf{pers}/1$. If `VRD56B25` is a *SSN*, then $\mathbf{pers}(\texttt{VRD56B25})$ denotes a person.
- If a person is identified by her *name* and *dateOfBirth*, we can introduce a function symbol $\mathbf{pers}/2$. Then $\mathbf{pers}(\texttt{Vardi}, \texttt{25/2/56})$ denotes a person.

unibz

Mapping assertions

Mapping assertions are used to extract the data from the DB to populate the ontology.

We make use of **variable terms**, which are like object terms, but with variables instead of values as arguments of the functions.

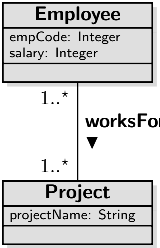
A **mapping assertion** between a database with schema $\mathcal{S}$ and a TBox $\mathcal{T}$ has the form

$$\Phi(\vec{x}) \leadsto \Psi(\vec{t}, \vec{y})$$

where

- Φ is an arbitrary SQL query of arity $n > 0$ over $\mathcal{S}$;
- Ψ is a conjunctive query over $\mathcal{T}$ of arity $n' > 0$ **without non-distinguished variables**;
- $\vec{x}$, $\vec{y}$ are variables, with $\vec{y} \subseteq \vec{x}$;
- $\vec{t}$ are variable terms of the form $\mathbf{f}(\vec{z})$, with $\mathbf{f} \in \Lambda$ and $\vec{z} \subseteq \vec{x}$.

Mapping assertions – Example



D$_1$[*SSN*: String, *PrName*: String]
Employees and Projects they work for

D$_2$[*Code*: String, *Salary*: Int]
Employee's code with salary

D$_3$[*Code*: String, *SSN*: String]
Employee's code with SSN

...

m_1: `SELECT SSN, PrName FROM D1` $\leadsto$ Employee(**pers**(*SSN*)), Project(**proj**(*PrName*)), projectName(**proj**(*PrName*), *PrName*), worksFor(**pers**(*SSN*), **proj**(*PrName*))

m_2: `SELECT SSN, Salary FROM D2, D3 WHERE D2.Code = D3.Code` $\leadsto$ Employee(**pers**(*SSN*)), salary(**pers**(*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

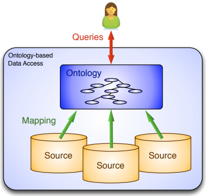
- Most popular RDB to RDF mapping language
- W3C Recommendation 27 Sep. 2012, `http://www.w3.org/TR/r2rml/`
- R2RML mappings are themselves expressed as RDF graphs and written in Turtle syntax.

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

unibz

Outline of Part 3

Ontology-based data access: Formalization



To formalize OBDA, we distinguish between the intensional and the extensional level information.

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

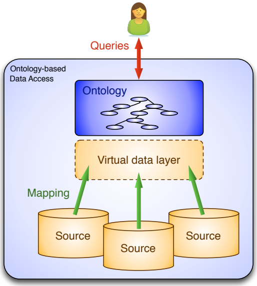
- $\mathcal{T}$ is a DL TBox providing the intensional level of an ontology.
- $\mathcal{S}$ is a (possibly federated) relational database schema for the data sources, possibly with constraints;
- $\mathcal{M}$ is a set of mapping assertions between $\mathcal{T}$ and $\mathcal{S}$.

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 $\mathcal{S}$.

Semantics of an OBDA system: Intuition

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



The data $\mathcal{D}$ and the mapping $\mathcal{M}$ define a **virtual data layer** $\mathcal{V}$, which behaves like a (virtual) ABox.

- Queries are answered w.r.t. $\mathcal{T}$ and $\mathcal{V}$.
- One aim is to avoid materializing the data of $\mathcal{V}$.
- Instead, the intensional information in $\mathcal{T}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{T}$ into queries formulated over $\mathcal{S}$.

OBDA vs. Ontology Based Query Answering (OBQA)

OBDA relies on OBQA to process queries w.r.t. the TBox $\mathcal{T}$, but in addition is concerned with efficiently dealing with the mapping $\mathcal{M}$.

OBDA should not be confused with OBQA.

Semantics of mappings

To formally define the semantics of an OBDA system $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, where $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, we first need to define the semantics of mappings.

Satisfaction of a mapping assertion with respect to a database

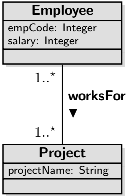
An interpretation $\mathcal{I}$ **satisfies** a mapping assertion $\Phi(\vec{x}) \leadsto \Psi(\vec{t}, \vec{y})$ in $\mathcal{M}$ **with respect to a database** $\mathcal{D}$ **for** $\mathcal{S}$, if for each tuple of values $\vec{v} \in \mathit{Eval}(\Phi, \mathcal{D})$, and for each ground atom in $\Psi[\vec{x}/\vec{v}]$, we have that:

- if the ground atom is $A(s)$, then $s^{\mathcal{I}} \in A^{\mathcal{I}}$.
- if the ground atom is $P(s_1, s_2)$, then $(s_1^{\mathcal{I}}, s_2^{\mathcal{I}}) \in P^{\mathcal{I}}$.

Intuitively, $\mathcal{I}$ **satisfies** $\Phi \leadsto \Psi$ w.r.t. $\mathcal{D}$ if all facts obtained by evaluating Φ over $\mathcal{D}$ and then propagating the answers to Ψ, hold in $\mathcal{I}$.

Note: $\mathit{Eval}(\Phi, \mathcal{D})$ denotes the result of evaluating Φ over the database $\mathcal{D}$.
$\Psi[\vec{x}/\vec{v}]$ denotes Ψ where each x_i has been substituted with v_i.

Semantics of mappings – Example



D_1:

SSN	*PrName*
23AB	optique
...	...

D_2:

Code	*Salary*
e23	1500
...	...

D_3:

Code	*SSN*
e23	23AB
...	...

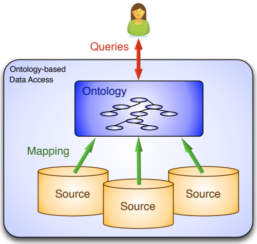
The following interpretation $\mathcal{I}$ satisfies the mapping assertions m_1 and m_2 with respect to the above database:

$$\mathcal{I} : \Delta_O^{\mathcal{I}} = \{\textbf{pers}(\texttt{23AB}), \textbf{proj}(\texttt{optique}), \ldots\}, \quad \Delta_V^{\mathcal{I}} = \{\texttt{optique}, 1500, \ldots\}$$
$$\mathsf{Employee}^{\mathcal{I}} = \{\textbf{pers}(\texttt{23AB}), \ldots\}, \quad \mathsf{Project}^{\mathcal{I}} = \{\textbf{proj}(\texttt{optique}), \ldots\},$$
$$\mathsf{projectName}^{\mathcal{I}} = \{(\textbf{proj}(\texttt{optique}), \texttt{optique}), \ldots\},$$
$$\mathsf{worksFor}^{\mathcal{I}} = \{(\textbf{pers}(\texttt{23AB}), \textbf{proj}(\texttt{optique})), \ldots\},$$
$$\mathsf{salary}^{\mathcal{I}} = \{(\textbf{pers}(\texttt{23AB}), 1500), \ldots\}$$

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

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

Semantics of an OBDA system



Model of an OBDA system

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

An OBDA system $\mathcal{O}$ is **satisfiable** if it admits at least one model.

Outline of Part 3

unibz

Answering queries over an OBDA system

In an OBDA system $\mathcal{O} = \langle \mathcal{P}, \mathcal{D} \rangle$, where $\mathcal{P} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$

- Queries are posed over the TBox $\mathcal{T}$.
- The data needed to answer queries is stored in the database $\mathcal{D}$.
- The mapping $\mathcal{M}$ is used to bridge the gap between $\mathcal{T}$ and $\mathcal{S}$.

Two approaches to exploit the mapping:

- bottom-up approach: simpler, but less efficient
- top-down approach: more sophisticated, but also more efficient

Note: Both approaches require to first **split** the TBox queries in the mapping assertions into their constituent atoms.

Splitting of mappings

A mapping assertion $\Phi \rightsquigarrow \Psi$, where the TBox query Ψ is constituted by the atoms X_1,...,X_k, can be split into several mapping assertions:

$$\Phi \rightsquigarrow X_1 \qquad \cdots \qquad \Phi \rightsquigarrow X_k$$

This is possible, since Ψ does not contain non-distinguished variables.

Example

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

is split into

m_1^1: `SELECT SSN, PrName FROM D1` $\rightsquigarrow$ Employee(**pers**(*SSN*))
m_1^2: `SELECT SSN, PrName FROM D1` $\rightsquigarrow$ Project(**proj**(*PrName*))
m_1^3: `SELECT SSN, PrName FROM D1` $\rightsquigarrow$ projectName(**proj**(*PrName*), *PrName*)
m_1^4: `SELECT SSN, PrName FROM D1` $\rightsquigarrow$ worksFor(**pers**(*SSN*), **proj**(*PrName*))

Bottom-up approach to query answering

Consists in a straightforward application of the mappings:

1. Propagate the data from $\mathcal{D}$ through $\mathcal{M}$, materializing an ABox $\mathcal{A}_{\mathcal{M},\mathcal{D}}$ (the constants in such an ABox are values and object terms).
2. Apply to $\mathcal{A}_{\mathcal{M},\mathcal{D}}$ and to the TBox $\mathcal{T}$, the satisfiability and query answering algorithms developed for *DL-Lite*$_{\mathcal{A}}$.

This approach has several drawbacks (hence is only theoretical):

- The technique is no more AC^0 in the data, since the ABox $\mathcal{A}_{\mathcal{M},\mathcal{D}}$ to materialize is in general polynomial in the size of the data.
- $\mathcal{A}_{\mathcal{M},\mathcal{D}}$ may be very large, and thus it may be infeasible to actually materialize it.
- Freshness of $\mathcal{A}_{\mathcal{M},\mathcal{D}}$ with respect to the underlying data source(s) may be an issue, and one would need to propagate source updates (cf. Data Warehousing).

Top-down approach to query answering

Consists of three steps:

1. **Reformulation:** Compute the perfect reformulation $q_{pr} = \mathit{PerfectRef}(q, \mathcal{T}_P)$ of the original query q, using the inclusion assertions of the TBox $\mathcal{T}$ (see later).

2. **Unfolding:** Compute from q_{pr} a new query q_{unf} by unfolding q_{pr} using (the split version of) the mappings $\mathcal{M}$.
   - Essentially, each atom in q_{pr} that unifies with an atom in Ψ is substituted with the corresponding query Φ over the database.
   - The unfolded query is such that $\mathit{Eval}(q_{unf}, \mathcal{D}) = \mathit{Eval}(q_{pr}, \mathcal{A}_{\mathcal{M},\mathcal{D}})$.

3. **Evaluation:** Delegate the evaluation of q_{unf} to the relational DBMS managing $\mathcal{D}$.

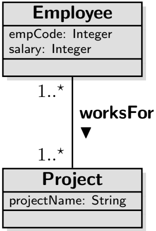
Unfolding

To unfold a query q_{pr} with respect to a set of mapping assertions:

1. For each non-split mapping assertion $\Phi_i(\vec{x}) \rightsquigarrow \Psi_i(\vec{t}, \vec{y})$:
    1. Introduce a **view symbol** Aux_i of arity equal to that of Φ_i.
    2. Add a **view definition** $\mathsf{Aux}_i(\vec{x}) \leftarrow \Phi_i(\vec{x})$.
2. For each split version $\Phi_i(\vec{x}) \rightsquigarrow X_j(\vec{t}, \vec{y})$ of a mapping assertion, introduce a **clause** $X_j(\vec{t}, \vec{y}) \leftarrow \mathsf{Aux}_i(\vec{x})$.
3. Obtain from q_{pr} in all possible ways queries q_{aux} defined over the view symbols Aux_i as follows:
    1. Find a most general unifier ϑ that unifies each atom $X(\vec{z})$ in the body of q_{pr} with the head of a clause $X(\vec{t}, \vec{y}) \leftarrow \mathsf{Aux}_i(\vec{x})$.
    2. Substitute each atom $X(\vec{z})$ with $\vartheta(\mathsf{Aux}_i(\vec{x}))$, i.e., with the body the unified clause to which the unifier ϑ is applied.
4. The unfolded query q_{unf} is the **union** of all queries q_{aux}, together with the view definitions for the predicates Aux_i appearing in q_{aux}.

unibz

Unfolding – Example



m_1: `SELECT SSN, PrName FROM D`$_1$ $\leadsto$ Employee(**pers**(*SSN*)), Project(**proj**(*PrName*)), projectName(**proj**(*PrName*), *PrName*), worksFor(**pers**(*SSN*), **proj**(*PrName*))

m_2: `SELECT SSN, Salary FROM D`$_2$`, D`$_3$ `WHERE D`$_2$`.Code = D`$_3$`.Code` $\leadsto$ Employee(**pers**(*SSN*)), salary(**pers**(*SSN*), *Salary*)

We define a view Aux_i for the source query of each mapping m_i.

For each (split) mapping assertion, we introduce a clause:

$$
\begin{array}{rcl}
\text{Employee}(\textbf{pers}(SSN)) & \leftarrow & \text{Aux}_1(SSN, PrName) \\
\text{projectName}(\textbf{proj}(PrName), PrName) & \leftarrow & \text{Aux}_1(SSN, PrName) \\
\text{Project}(\textbf{proj}(PrName)) & \leftarrow & \text{Aux}_1(SSN, PrName) \\
\text{worksFor}(\textbf{pers}(SSN), \textbf{proj}(PrName)) & \leftarrow & \text{Aux}_1(SSN, PrName) \\
\text{Employee}(\textbf{pers}(SSN)) & \leftarrow & \text{Aux}_2(SSN, Salary) \\
\text{salary}(\textbf{pers}(SSN), Salary) & \leftarrow & \text{Aux}_2(SSN, Salary)
\end{array}
$$

unibz

Unfolding – Example (cont'd)

Query over ontology: employees who work for optique and their salary:
$q(e, s) \leftarrow \mathsf{Employee}(e), \mathsf{salary}(e, s), \mathsf{worksFor}(e, p), \mathsf{projectName}(p, \mathtt{optique})$

A unifier between the atoms in q and the clause heads is:

$$\vartheta(e) = \mathbf{pers}(\mathit{SSN}) \qquad \vartheta(s) = \mathit{Salary}$$
$$\vartheta(\mathit{PrName}) = \mathtt{optique} \qquad \vartheta(p) = \mathbf{proj}(\mathtt{optique})$$

After applying ϑ to q, we obtain:

$$\begin{aligned} q(\mathbf{pers}(\mathit{SSN}), \mathit{Salary}) \leftarrow\ & \mathsf{Employee}(\mathbf{pers}(\mathit{SSN})),\ \mathsf{salary}(\mathbf{pers}(\mathit{SSN}), \mathit{Salary}),\\ & \mathsf{worksFor}(\mathbf{pers}(\mathit{SSN}), \mathbf{proj}(\mathtt{optique})),\\ & \mathsf{projectName}(\mathbf{proj}(\mathtt{optique}), \mathtt{optique}) \end{aligned}$$

Substituting the atoms with the bodies of the unified clauses, we obtain:

$$\begin{aligned} q(\mathbf{pers}(\mathit{SSN}), \mathit{Salary}) \leftarrow\ & \mathsf{Aux}_1(\mathit{SSN}, \mathtt{optique}),\ \mathsf{Aux}_2(\mathit{SSN}, \mathit{Salary}),\\ & \mathsf{Aux}_1(\mathit{SSN}, \mathtt{optique}),\ \mathsf{Aux}_1(\mathit{SSN}, \mathtt{optique}) \end{aligned}$$

unibz

Exponential blowup in the unfolding

When there are multiple mapping assertions for each atom, the unfolded query may be exponential in the original one.

Consider a query: $q(y) \leftarrow A_1(y), A_2(y), \ldots, A_n(y)$

and the mappings: $m_i^1\colon \Phi_i^1(x) \leadsto A_i(\mathbf{f}(x))$ (for $i \in \{1, \ldots, n\}$)
$m_i^2\colon \Phi_i^2(x) \leadsto A_i(\mathbf{f}(x))$

We add the view definitions: $\mathsf{Aux}_i^j(x) \leftarrow \Phi_i^j(x)$
and introduce the clauses: $A_i(\mathbf{f}(x)) \leftarrow \mathsf{Aux}_i^j(x)$ (for $i \in \{1, \ldots, n\}$, $j \in \{1, 2\}$).

There is a single unifier, namely $\vartheta(y) = \mathbf{f}(x)$, but each atom $A_i(y)$ in the query unifies with the head of two clauses.

Hence, we obtain one unfolded query

$$q(\mathbf{f}(x)) \leftarrow \mathsf{Aux}_1^{j_1}(x), \mathsf{Aux}_2^{j_2}(x), \ldots, \mathsf{Aux}_n^{j_n}(x)$$

for each possible combination of $j_i \in \{1, 2\}$, for $i \in \{1, \ldots, n\}$.
Hence, we obtain 2^n **unfolded queries**.

Computational complexity of query answering

From the top-down approach to query answering, and the complexity results for *DL-Lite*, we obtain the following result.

Theorem

Query answering in a *DL-Lite* OBDM system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ is

1. NP-**complete** in the size of the query.
2. PTime in the size of the **TBox** $\mathcal{T}$ and the **mappings** $\mathcal{M}$.
3. AC^0 in the size of the **database** $\mathcal{D}$.

Note: The AC^0 result is a consequence of the fact that query answering in such a setting can be reduced to evaluating an SQL query over the relational database.

unibz

Implementation of top-down approach to query answering

To implement the top-down approach, we need to generate an SQL query.

We can follow different strategies:

1. Substitute each view predicate in the unfolded queries with the corresponding SQL query over the source:
   + \+ joins are performed on the DB attributes;
   + \+ does not generate doubly nested queries;
   + – the number of unfolded queries may be exponential.

2. Construct for each atom in the original query a new view. This view takes the union of all SQL queries corresponding to the view predicates, and constructs also the Skolem terms:
   + \+ avoids exponential blow-up of the resulting query, since the union (of the queries coming from multiple mappings) is done before the joins;
   + – joins are performed on Skolem terms;
   + – generates doubly nested queries.

Which method is better, depends on various parameters.

unibz

Towards answering arbitrary SQL queries

- We have seen that answering full SQL (i.e., FOL) queries is undecidable.
- However, we can treat the answers to an UCQ, as "knowledge", and perform further computations on that knowledge.
- This corresponds to applying a knowledge operator to UCQs that are embedded into an arbitrary SQL query (EQL queries) [Calvanese *et al.*, 2007b]
  - The UCQs are answered according to the certain answer semantics.
  - The SQL query is evaluated on the facts returned by the UCQs.
- The approach can be implemented by rewriting the UCQs and embedding the rewritten UCQs into SQL.
- The user "sees" arbitrary SQL queries, but these SQL queries are evaluated according to a weakened semantics.

unibz

Outline of Part 3

unibz

Experimentations and experiences

Several experimentations (in rough chronological order):

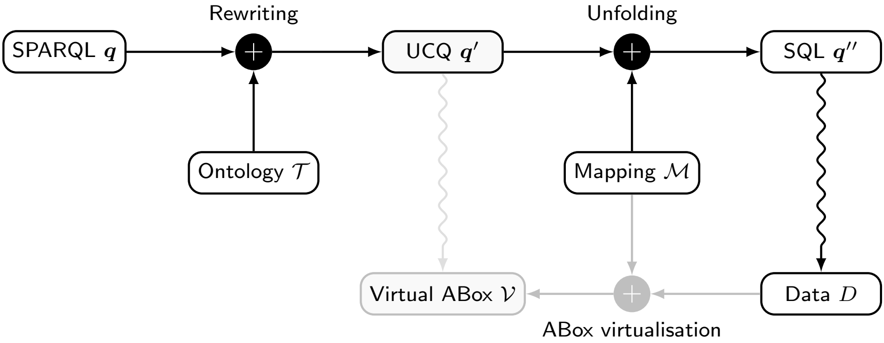
- Monte dei Paschi di Siena (led by Sapienza Univ. of Rome)
- Selex: world leading radar producer
- National Accessibility Portal of South Africa
- Horizontal Gene Transfer data and ontology
- Stanford's "Resource Index" comprising 200 ontologies from BioPortal
- Norwegian Petroleum Directorate (NPD) FactPages (within Optique)
- Benchmarking on (partially) 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'':

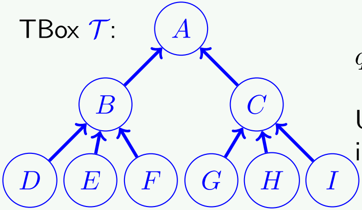
- Existentials: Sub-queries of q with **existentially quantified variables** might lead in general to exponentially large rewritings.
- Hierarchies: Concepts / roles occurring in the query q can have **many subconcepts / subroles** according to $\mathcal{T}$, which all have to be included in the rewriting 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''.

unibz

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 [Calvanese *et al.*, 2007c].

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**.

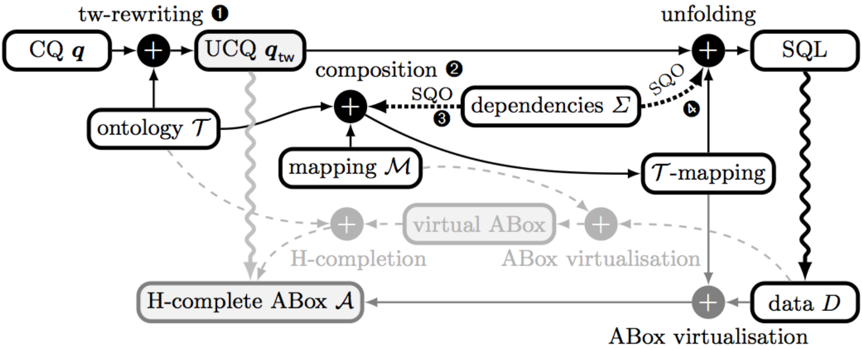
Observe:

- 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 of Part 3

unibz

Main publications

Most of the results presented in this course have been published:

- Reasoning and query answering in *DL-Lite*: [Calvanese *et al.*, 2005; Calvanese *et al.*, 2006b; Calvanese *et al.*, 2007c; Calvanese *et al.*, 2007a; Artale *et al.*, 2009; Calvanese *et al.*, 2013b]
- Mapping to data sources and OBDA: [Calvanese *et al.*, 2006a; Calvanese *et al.*, 2008a; Poggi *et al.*, 2008a]
- Connection between description logics and conceptual modeling formalisms: [Calvanese *et al.*, 1998; Berardi *et al.*, 2005; Artale *et al.*, 2007; Calvanese *et al.*, 2009b]
- Tool descriptions: [Poggi *et al.*, 2008b; Rodríguez-Muro and Calvanese, 2008; Calvanese *et al.*, 2011; Rodriguez-Muro and Calvanese, 2012]
- Optimization techniques: [Rodriguez-Muro *et al.*, 2013; Kontchakov *et al.*, 2014]
- Case studies: [Keet *et al.*, 2008; Savo *et al.*, 2010; Calvanese *et al.*, 2013a]

A summary of many of the presented results and techniques, with detailed proofs is given in [Calvanese *et al.*, 2009a].

Query rewriting for more expressive ontology languages

These result have then been extended to more expressive ontology languages, using different techniques:

- In [Artale *et al.*, 2009] various *DL-Lite* extensions are considered, providing a comprehensive treatment of the expressiveness/complexity trade-off for the *DL-Lite* family and related logics:
  - number restrictions besides functionality;
  - conjunction on the left-hand side of inclusions (horn logics);
  - boolean constructs;
  - constraints on roles, such as (ir)reflexivity, (a)symmetry, transitivity;
  - presence and absence of the unique name assumption.
- Alternative query rewriting techniques based on resolution, and applicable also to more expressive logics (leading to recursive rewritings) [Pérez-Urbina *et al.*, 2010].
- Query rewriting techniques for database inspired constraint languages [Calì *et al.*, 2009a; Calì *et al.*, 2009b; Calì *et al.*, 2012; Gottlob *et al.*, 2014].

Further theoretical work

The results presented in this course have also inspired additional work relevant for ontology-based data access:

- We have considered mainly query answering. However, several other ontology-based services are of importance:
  - write-also access: updating a data source through an ontology [De Giacomo *et al.*, 2009; Calvanese *et al.*, 2010; Zheleznyakov *et al.*, 2010]
  - modularity and minimal module extraction [Kontchakov *et al.*, 2008; Kontchakov *et al.*, 2009]
  - privacy aware data access [Calvanese *et al.*, 2008b]
  - meta-level reasoning and query answering, a la RDFS [De Giacomo *et al.*, 2008]
  - provenance and explanation [Borgida *et al.*, 2008]
- Reasoning with respect to finite models only [Rosati, 2008].
- We have dealt only with the static aspects of information systems. However a crucial issue is how to deal with **dynamic aspects**. See also work carried out in the EU project ACSI.

Work on most of these issues is still ongoing.

Further practical and experimental work

The theoretical results indicate a good computational behaviour in the size of the data. However, performance is a critical issue in practice:

- The rewriting consists of a large number of CQs. Query containment can be used to prune the rewriting. This is already implemented in -ontop-system, but requires further optimizations.
- The SQL queries generated by the mapping unfolding are not easy to process by the DBMS engine (e.g., they may contain complex joins on skolem terms computed on the fly).
  Different mapping unfolding strategies have a strong impact on computational complexity. Experimentation is ongoing to assess the tradeoff.
- Further extensive experimentations are ongoing:
  - on artificially generated data;
  - on real-world use cases.
- An OBDA benchmarking suite is under development [Calvanese *et al.*, 2014; Lanti *et al.*, 2015].

Acknowledgements

Many people are/were involved in various aspects of this work:

- Alessandro Artale[1]
- Benjamin Cogrel[1]
- Giuseppe De Giacomo[2]
- Sarah Komla Ebri[1]
- Roman Kontchakov[3]
- Davide Lanti[1]
- Domenico Lembo[2]
- Maurizio Lenzerini[2]
- Antonella Poggi[2]
- Martin Rezk[1]
- Mariano Rodriguez Muro[1,4]
- Riccardo Rosati[2]
- Domenico Fabio Savo[2]
- Mindaugas Slusnys[1]
- Guohui Xiao[1]
- Michael Zakharyaschev[3]

[1] Free University of Bozen-Bolzano
[2] Sapienza Università di Roma
[3] Birkbeck College London
[4] IBM Research Watson

Thanks go also to many developers!

The development work on -ontop- and the experimentation are currently carried out within the EU Large Scale Integrating Project *Optique* (*Scalable End-user Access to Big Data*), grant n. FP7-318338, http://www.optique-project.eu/

ontop

Optique

unibz

References I

[Artale *et al.*, 2007] Alessandro Artale, Diego Calvanese, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev.
Reasoning over extended ER models.
In *Proc. of the 26th Int. Conf. on Conceptual Modeling (ER)*, volume 4801 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2007.

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

[Berardi *et al.*, 2005] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo.
Reasoning on UML class diagrams.
Artificial Intelligence, 168(1–2):70–118, 2005.

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

References II

[Calì *et al.*, 2009a] 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.

[Calì *et al.*, 2009b] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz.
A general Datalog-based framework for tractable query answering over ontologies.
In *Proc. of the 28th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, pages 77–86, 2009.

[Calì *et al.*, 2012] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz.
A general Datalog-based framework for tractable query answering over ontologies.
J. of Web Semantics, 14:57–83, 2012.

[Calvanese *et al.*, 1998] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi.
Description logics for conceptual data modeling.
In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publishers, 1998.

References III

[Calvanese *et al.*, 2005] Diego Calvanese, 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.

[Calvanese *et al.*, 2006a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati.
Linking data to ontologies: The description logic $\mathit{DL\text{-}Lite}_a$.
In *Proc. of the 2nd Int. Workshop on OWL: Experiences and Directions (OWLED)*, volume 216 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2006.

[Calvanese *et al.*, 2006b] Diego Calvanese, 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.

References IV

[Calvanese *et al.*, 2007a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
Can OWL model football leagues?
In *Proc. of the 3rd Int. Workshop on OWL: Experiences and Directions (OWLED)*, volume 258 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2007.

[Calvanese *et al.*, 2007b] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
EQL-Lite: Effective first-order query processing in description logics.
In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 274–279, 2007.

[Calvanese *et al.*, 2007c] Diego Calvanese, 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.

unibz

References V

[Calvanese *et al.*, 2008a] Diego Calvanese, 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.

[Calvanese *et al.*, 2008b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati.
View-based query answering over description logic ontologies.
In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pages 242–251, 2008.

[Calvanese *et al.*, 2009a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodríguez-Muro, and Riccardo Rosati.
Ontologies and databases: The *DL-Lite* approach.
In Sergio Tessaris and Enrico Franconi, editors, *Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.

unibz

References VI

[Calvanese *et al.*, 2009b] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
Conceptual modeling for data integration.
In Alex T. Borgida, Vinay Chaudhri, Paolo Giorgini, and Eric Yu, editors, *Conceptual Modeling: Foundations and Applications – Essays in Honor of John Mylopoulos*, volume 5600 of *Lecture Notes in Computer Science*, pages 173–197. Springer, 2009.

[Calvanese *et al.*, 2010] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov.
Updating ABoxes in *DL-Lite*.
In *Proc. of the 4th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW)*, volume 619 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 3.1–3.12, 2010.

[Calvanese *et al.*, 2011] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.
The Mastro system for ontology-based data access.
Semantic Web J., 2(1):43–53, 2011.

References VII

[Calvanese *et al.*, 2013a] D. Calvanese, M. Giese, P. Haase, I. Horrocks, T. Hubauer, Y. Ioannidis, E. Jiménez-Ruiz, E. Kharlamov, H. Kllapi, J. Klüwer, M. Koubarakis, S. Lamparter, R. Möller, C. Neuenstadt, T. Nordtveit, Ö. Özcep, M. Rodriguez-Muro, M. Roshchin, F. Savo, M. Schmidt, A. Soylu, A. Waaler, and D. Zheleznyakov.

Optique: OBDA solution for big data.

In *Revised Selected Papers of ESWC 2013 Satellite Events*, volume 7955 of *Lecture Notes in Computer Science*, pages 293–295. Springer, 2013.

[Calvanese *et al.*, 2013b] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Data complexity of query answering in description logics.

Artificial Intelligence, 195:335–360, 2013.

[Calvanese *et al.*, 2014] Diego Calvanese, Davide Lanti, Martin Rezk, Mindaugas Slusnys, and Guohui Xiao.

A scalable benchmark for OBDA systems: Preliminary report.

In *Proc. of the 3rd Int. Workshop on OWL Reasoner Evaluation (ORE)*, volume 1207 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2014.

References VIII

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

[De Giacomo *et al.*, 2009] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati.
On instance-level update and erasure in description logic ontologies.
J. of Logic and Computation, Special Issue on Ontology Dynamics, 19(5):745–770, 2009.

[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.

[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 IX

[Keet *et al.*, 2008] C. Maria Keet, Ronell Alberts, Aurona Gerber, and Gibson Chimamiwa.
Enhancing web portals with Ontology-Based Data Access: the case study of South Africa's Accessibility Portal for people with disabilities.
In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED)*, volume 432 of *CEUR Electronic Workshop Proceedings*, http://ceur-ws.org/, 2008.

[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)*, pages 275–285, 2012.

[Kontchakov *et al.*, 2008] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev.
Can you tell the difference between *DL-Lite* ontologies?
In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pages 285–295, 2008.

[Kontchakov *et al.*, 2009] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev.
Minimal module extraction from DL-Lite ontologies using QBF solvers.
In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 836–840, 2009.

References X

[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)*, pages 247–257, 2010.

[Kontchakov *et al.*, 2014] Roman Kontchakov, Martin Rezk, Mariano Rodriguez-Muro, Guohui Xiao, and Michael Zakharyaschev.
Answering SPARQL queries over databases under OWL 2 QL entailment regime.
In *Proc. of the 13th Int. Semantic Web Conf. (ISWC)*, volume 8796 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2014.

[Lanti *et al.*, 2015] Davide Lanti, Martin Rezk, Guohui Xiao, and Diego Calvanese.
The NPD benchmark: Reality check for OBDA systems.
In *Proc. of the 18th Int. Conf. on Extending Database Technology (EDBT)*, 2015.

unibz

References XI

[Motik *et al.*, 2009] Boris Motik, Achille Fokoue, Ian Horrocks, Zhe Wu, Carsten Lutz, and Bernardo Cuenca Grau.
OWL Web Ontology Language profiles.
W3C Recommendation, World Wide Web Consortium, October 2009.
Available at `http://www.w3.org/TR/owl-profiles/`.

[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.

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

[Poggi *et al.*, 2008b] Antonella Poggi, Mariano Rodríguez-Muro, and Marco Ruzzi.
Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé.
In Kendall Clark and Peter F. Patel-Schneider, editors, *Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED DC)*, 2008.

References XII

[Rodríguez-Muro and Calvanese, 2008] Mariano Rodríguez-Muro and Diego Calvanese.
Towards an open framework for ontology based data access with Protégé and DIG 1.1.
In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED)*, volume 432 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2008.

[Rodriguez-Muro and Calvanese, 2012] Mariano Rodriguez-Muro and Diego Calvanese.
High performance query answering over *DL-Lite* ontologies.
In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pages 308–318, 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.

[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)*, pages 290–300, 2010.

References XIII

[Rosati, 2008] Riccardo Rosati.
Finite model reasoning in *DL-Lite*.
In *Proc. of the 5th European Semantic Web Conf. (ESWC)*, 2008.

[Savo *et al.*, 2010] Domenico Fabio Savo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodríguez-Muro, Vittorio Romagnoli, Marco Ruzzi, and Gabriele Stella.
Mastro at work: Experiences on ontology-based data access.
In *Proc. of the 23rd Int. Workshop on Description Logic (DL)*, volume 573 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 20–31, 2010.

[Zheleznyakov *et al.*, 2010] Dmitriy Zheleznyakov, Diego Calvanese, Evgeny Kharlamov, and Werner Nutt.
Updating TBoxes in *DL-Lite*.
In *Proc. of the 23rd Int. Workshop on Description Logic (DL)*, volume 573 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 102–113, 2010.