Knowledge Representation and Ontologies

Part 4: Ontology Based Data Access

Diego Calvanese

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2011/2012

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

Part 4

Ontology-based data access

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics
   - Basic features of *DL-Lite*
   - Syntax and semantics of *DL-Lite*
   - Identification assertions in *DL-Lite*
   - Members of the *DL-Lite* family
   - Properties of *DL-Lite*
2. Linking ontologies to relational data
   - The impedance mismatch problem
   - Ontology-Based Data Access systems
   - Query answering in Ontology-Based Data Access systems
   - The QUEST system for Ontology-Based Data Access
3. Further work and references

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics
   - Basic features of *DL-Lite*
   - Syntax and semantics of *DL-Lite*
   - Identification assertions in *DL-Lite*
   - Members of the *DL-Lite* family
   - Properties of *DL-Lite*
2. Linking ontologies to relational data
3. Further work and references

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics
   - Basic features of *DL-Lite*
   - Syntax and semantics of *DL-Lite*
   - Identification assertions in *DL-Lite*
   - Members of the *DL-Lite* family
   - Properties of *DL-Lite*
2. Linking ontologies to relational data
3. Further work and references

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).
- Is equipped with identification assertions.
- 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).

Outline of Part 4

Syntax of the $DL\text{-}Lite_{\mathcal{A}}$ description language

- Role expressions:
  - atomic role: P
  - basic role: $Q ::= P \mid P^-$
  - arbitrary role: $R ::= Q \mid \neg Q$ (to express disjointness)
- Concept expressions:
  - 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:
  - atomic attribute: U
  - arbitrary attribute: $V := U \mid \neg U$ (to express disjointness)
- Value-domain expressions:
  - attribute range: $\rho(U)$
  - RDF datatypes: T_i
  - top domain: $\top_D$

Semantics of $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ – 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 $c^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$	Value constants Γ_V $d^{\mathcal{I}} = \mathit{val}(d)$ given a priori
Unary predicates	Concept C $C^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}}$	RDF datatype T_i $T_i^{\mathcal{I}} \subseteq \Delta_V^{\mathcal{I}}$ given a priori
Binary predicates	Role R $R^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}$	Attribute V $V^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$

Semantics of the $DL\text{-}Lite_{\mathcal{A}}$ 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	$\neg Q$	¬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	$\exists Q$	∃child$^-$	$\{o \mid \exists o'. (o, o') \in Q^{\mathcal{I}}\}$
concept negation	$\neg B$	¬∃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	$\neg U$	¬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'`	$\mathit{val}(d) \in \Delta_V^{\mathcal{I}}$ (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

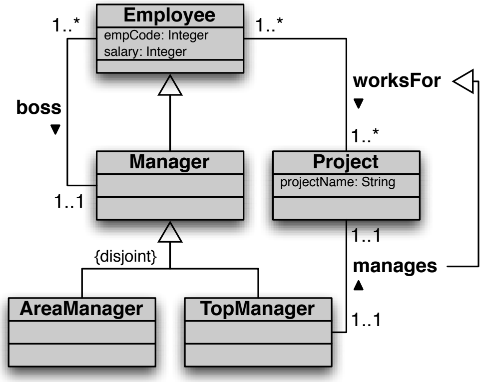
ABox assertions: $A(c)$, $P(c, c')$, $U(c, d)$,
where c, c' are object constants and d is a value constant

Semantics of the $DL\text{-}Lite_{\mathcal{A}}$ 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.it

DL-Lite$_\mathcal{A}$ – Example

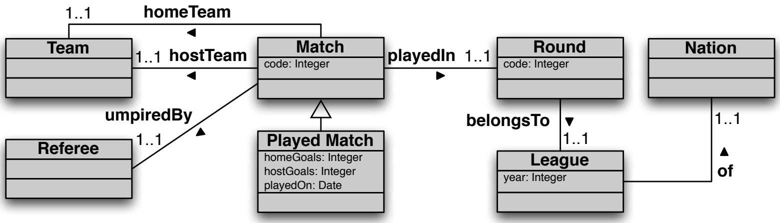


$$
\begin{array}{rcl}
\mathsf{Manager} & \sqsubseteq & \mathsf{Employee} \\
\mathsf{AreaManager} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{TopManager} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{AreaManager} & \sqsubseteq & \neg\mathsf{TopManager} \\
\mathsf{Employee} & \sqsubseteq & \delta(\mathsf{empCode}) \\
\delta(\mathsf{empCode}) & \sqsubseteq & \mathsf{Employee} \\
\rho(\mathsf{empCode}) & \sqsubseteq & \mathtt{xsd:int} \\
\multicolumn{3}{c}{(\mathbf{funct}\ \mathsf{empCode})} \\
\multicolumn{3}{c}{(\mathbf{id}\ \mathsf{Employee}\ \mathsf{empCode})} \\
\exists\mathsf{worksFor} & \sqsubseteq & \mathsf{Employee} \\
\exists\mathsf{worksFor}^- & \sqsubseteq & \mathsf{Project} \\
\mathsf{Employee} & \sqsubseteq & \exists\mathsf{worksFor} \\
\mathsf{Project} & \sqsubseteq & \exists\mathsf{worksFor}^- \\
\multicolumn{3}{c}{(\mathbf{funct}\ \mathsf{manages})} \\
\multicolumn{3}{c}{(\mathbf{funct}\ \mathsf{manages}^-)} \\
\mathsf{manages} & \sqsubseteq & \mathsf{worksFor} \\
 & \vdots &
\end{array}
$$

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

Outline of Part 4

Identification assertions – Example



What we would like to additionally capture:

1. No two leagues with the same year and the same nation exist
2. Within a certain league, the code associated to a round is unique
3. Every match is identified by its code within its round
4. Every referee can umpire at most one match in the same round
5. No team can be the home team of more than one match per round
6. No team can be the host team of more than one match per round

Identification assertions – Example (cont'd)

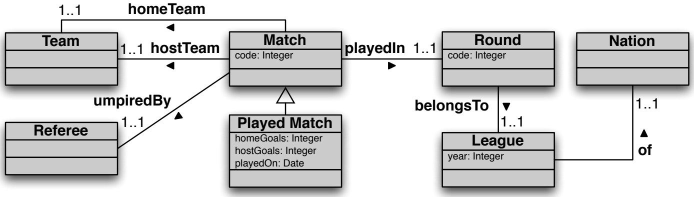
$\mathsf{League} \sqsubseteq \exists \mathsf{of}$
$\exists \mathsf{of} \sqsubseteq \mathsf{League}$
$\exists \mathsf{of}^- \sqsubseteq \mathsf{Nation}$
$\mathsf{Round} \sqsubseteq \exists \mathsf{belongsTo}$
$\exists \mathsf{belongsTo} \sqsubseteq \mathsf{Round}$
$\exists \mathsf{belongsTo}^- \sqsubseteq \mathsf{League}$
$\mathsf{Match} \sqsubseteq \exists \mathsf{playedIn}$
...

$\mathsf{PlayedMatch} \sqsubseteq \mathsf{Match}$
$\mathsf{Match} \sqsubseteq \delta(\mathsf{code})$
$\mathsf{Round} \sqsubseteq \delta(\mathsf{code})$
$\mathsf{PlayedMatch} \sqsubseteq \delta(\mathsf{playedOn})$
...
$\rho(\mathsf{playedOn}) \sqsubseteq \texttt{xsd:date}$
$\rho(\mathsf{code}) \sqsubseteq \texttt{xsd:int}$
...

(**funct** of)
(**funct** belongsTo)
(**funct** playedIn)
(**funct** homeTeam)
(**funct** hostTeam)
(**funct** umpiredBy)
(**funct** code)
(**funct** year)
(**funct** homeGoals)
(**funct** hostGoals)
(**funct** playedOn)

unibz.it

Identification assertions – Example (cont'd)



1. No two leagues with the same year and the same nation exist
2. Within a certain league, the code associated to a round is unique
3. Every match is identified by its code within its round
4. Every referee can umpire at most one match in the same round
5. No team can be the home team of more than one match per round
6. No team can be the host team of more than one match per round

(**id** League of, year)
(**id** Round belongsTo, code)
(**id** Match playedIn, code)
(**id** Match umpiredBy, playedIn)
(**id** Match homeTeam, playedIn)
(**id** Match hostTeam, playedIn)

Semantics of identification assertions

Let $(\textbf{id}\ B\ I_1, \ldots, I_n)$ be an identification assertion in a *DL-Lite*$_\mathcal{A}$ TBox.

An interpretation $\mathcal{I}$ satisfies such an assertion if for all $o_1, o_2 \in B^\mathcal{I}$ and for all objects or values $u_1, \ldots, u_n$, we have that

$$(o_1, u_j) \in I_j^\mathcal{I} \text{ and } (o_2, u_j) \in I_j^\mathcal{I}, \text{ for } j \in \{1, \ldots, n\}, \text{ implies that } o_1 = o_2.$$

In other words, the instance o_i of B is identified by the tuple $(u_1, \ldots, u_n)$ of objects or values to which it is connected via $I_1, \ldots, I_n$, respectively.

Note: the roles or attributes I_j are not required to be functional or mandatory.

The above definition of semantics implies that, in the case where an instance $o \in B^\mathcal{I}$ is connected by means of $I_j^\mathcal{I}$ to a set $u_j^1, \ldots, u_j^k$ of objects (or values), it is each single u_j^h that contributes to the identification of o, and not the whole set $\{u_j^1, \ldots, u_j^k\}$.

Outline of Part 4

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

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

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

- $\textit{DL-Lite}_{\mathcal{F}}$: Allows for functionality assertions, but does not allow for role inclusion assertions.
- $\textit{DL-Lite}_{\mathcal{R}}$: Allows for role inclusion assertions, but does not allow for functionality assertions.

In both $\textit{DL-Lite}_{\mathcal{F}}$ and $\textit{DL-Lite}_{\mathcal{R}}$ we do not consider data values (and hence drop value domains and attributes).

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

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics
   - Basic features of *DL-Lite*
   - Syntax and semantics of *DL-Lite*
   - Identification assertions in *DL-Lite*
   - Members of the *DL-Lite* family
   - Properties of *DL-Lite*
2. Linking ontologies to relational data
3. Further work and references

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	$(\textbf{id } A\ P, \ldots, P'^-, \ldots, U, \ldots)$

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 $\begin{array}{l} B \sqsubseteq C_1 \\ B \sqsubseteq C_2 \end{array}$

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

Properties of $\textit{DL-Lite}_\mathcal{F}$

$\textit{DL-Lite}_\mathcal{F}$ does **not** enjoy the **finite model property**.

Example

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

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

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.

Properties of *DL-Lite*$_{\mathcal{R}}$

- *DL-Lite*$_{\mathcal{R}}$ **does enjoy the finite model property**. Hence, reasoning w.r.t. finite models is the same as reasoning w.r.t. arbitrary models.

- With role inclusion assertions, we can simulate **qualified existential quantification** in the rhs of an inclusion assertion $A_1 \sqsubseteq \exists Q.A_2$.

  To do so, we introduce a new role Q_{A_2} and:
  - the role inclusion assertion $Q_{A_2} \sqsubseteq Q$
  - the concept inclusion assertions: $$\begin{array}{rcl} A_1 & \sqsubseteq & \exists Q_{A_2} \\ \exists Q_{A_2}^- & \sqsubseteq & A_2 \end{array}$$

  In this way, we can consider $\exists Q.A$ in the right-hand side of an inclusion assertion as an abbreviation.

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

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

The OWL 2 QL Profile

OWL 2 defines three **profiles**: OWL 2 QL, OWL 2 EL, OWL 2 RL.

- 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."

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

1. We have seen that *DL-Lite*$_\mathcal{A}$ can capture the essential features of prominent conceptual modeling formalisms.
2. In the following, we will analyze reasoning in *DL-Lite*, and establish the following characterization of its computational properties:
   - 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).
   - Query answering for CQs and UCQs is:
     - PTIME in the size of the TBox.
     - AC^0 in the size of the ABox.
     - Exponential in the size of the **query** (NP-complete).
       Bad? ...not really, this is exactly as in relational DBs.
3. We will also see that *DL-Lite* is essentially the maximal DL enjoying these nice computational properties.

From (1), (2), and (3) we get that:

DL-Lite is a representation formalism that is very well suited to underlie ontology-based data management systems.

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics

2. Linking ontologies to relational data
   - The impedance mismatch problem
   - Ontology-Based Data Access systems
   - Query answering in Ontology-Based Data Access systems
   - The QUEST system for Ontology-Based Data Access

3. Further work and references

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics
2. Linking ontologies to relational data
   - The impedance mismatch problem
   - Ontology-Based Data Access systems
   - Query answering in Ontology-Based Data Access systems
   - The QUEST system for Ontology-Based Data Access
3. Further work and references

Managing ABoxes

In the traditional DL setting, it is assumed that the data is maintained in the ABox of the ontology:

- The ABox is perfectly compatible with the TBox:
  - the vocabulary of concepts, roles, and attributes is the one used in the TBox.
  - The ABox "stores" abstract objects, and these objects and their properties are those returned by queries over the ontology.
- There may be different ways to manage the ABox from a physical point of view:
  - Description Logics reasoners maintain the ABox is main-memory data structures.
  - When an ABox becomes large, managing it in secondary storage may be required, but this is again handled directly by the reasoner.

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**.

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.

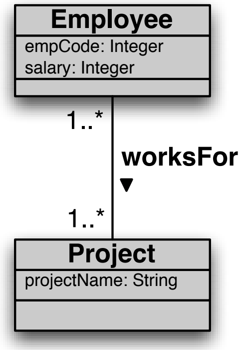
Note: the **ABox** is only **virtual**, and the objects are not materialized.

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).

Impedance mismatch – Example



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

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

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

$D_3[Code\colon \text{String}, SSN\colon \text{String}]$
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 $\texttt{VRD56B25}$ 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.

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.

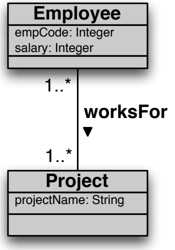
Def.: A **mapping assertion** between a database $\mathcal{D}$ and a TBox $\mathcal{T}$ has the form

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

where

- Φ is an arbitrary SQL query of arity $n > 0$ over $\mathcal{D}$;
- 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.it

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics

2. Linking ontologies to relational data
   - The impedance mismatch problem
   - Ontology-Based Data Access systems
   - Query answering in Ontology-Based Data Access systems
   - The QUEST system for Ontology-Based Data Access

3. Further work and references

Ontology-Based Data Access System

The mapping assertions are a crucial part of an Ontology-Based Data Access System.

Def.: Ontology-Based Data Access System

is a triple $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$, where

- $\mathcal{T}$ is a TBox.
- $\mathcal{D}$ is a relational database.
- $\mathcal{M}$ is a set of mapping assertions between $\mathcal{T}$ and $\mathcal{D}$.

Semantics of mappings

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

Def.: 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** $\mathcal{D}$, 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 an OBDA system

Def.: Model of an OBDA system

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

- $\mathcal{I}$ is a model of $\mathcal{T}$;
- $\mathcal{I}$ 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 4

Answering queries over an OBDA system

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

- 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 $\mathcal{D}$.

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

$$\Phi \leadsto X_1 \qquad \cdots \qquad \Phi \leadsto X_k$$

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

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*))

is split into

m_1^1: `SELECT SSN, PrName FROM D`$_1$ $\leadsto$ Employee(**pers**(*SSN*))
m_1^2: `SELECT SSN, PrName FROM D`$_1$ $\leadsto$ Project(**proj**(*PrName*))
m_1^3: `SELECT SSN, PrName FROM D`$_1$ $\leadsto$ projectName(**proj**(*PrName*), *PrName*)
m_1^4: `SELECT SSN, PrName FROM D`$_1$ $\leadsto$ 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 $\mathit{Eval}(q_{unf}, \mathcal{D}) = \mathit{Eval}(q_{pr}, \mathcal{A}_{\mathcal{M},\mathcal{D}})$.

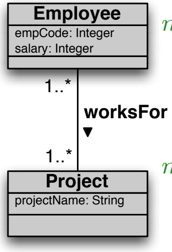
3. **Evaluation:** Delegate the evaluation of q_{unf} to the relational DBMS managing $\mathcal{D}$.

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** 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** $X_j(\vec{t}, \vec{y}) \leftarrow$ 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 $X(\vec{t}, \vec{y}) \leftarrow$ Aux$_i(\vec{x})$.
   2. Substitute each atom $X(\vec{z})$ with $\vartheta($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}.

Unfolding – Example



m_1: `SELECT SSN, PrName FROM` D_1 $\leadsto$ $\mathsf{Employee}(\mathbf{pers}(SSN))$, $\mathsf{Project}(\mathbf{proj}(PrName))$, $\mathsf{projectName}(\mathbf{proj}(PrName), PrName)$, $\mathsf{worksFor}(\mathbf{pers}(SSN), \mathbf{proj}(PrName))$

m_2: `SELECT SSN, Salary FROM` D_2, D_3 `WHERE` D_2`.Code =` D_3`.Code` $\leadsto$ $\mathsf{Employee}(\mathbf{pers}(SSN))$, $\mathsf{salary}(\mathbf{pers}(SSN), Salary)$

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

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

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

Unfolding – Example (cont'd)

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

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

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

After applying ϑ to q, we obtain:

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

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

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

unibz.it

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

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.

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.
Experiments have shown that (1) behaves better in most cases.

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.

Outline of Part 4

The Quest system

- Quest is a tool for representing and reasoning over ontologies of the *DL-Lite* family.
- The basic functionality it offers is query answering of UCQs
- Query answering is also at the basis of
  - ontology satisfiability;
  - intensional reasoning services: concept/role subsumption and disjunction, concept/role satisfiability.

  These functionalities will be made available natively in future versions.
- Reasoning services are highly optimized.
- Can be used with internal and external DBMS (includes drivers for various commercial and non-commercial DBMSs.
- Implemented in Java as an open source project.

Outline of Part 4

1. The *DL-Lite* family of tractable Description Logics
2. Linking ontologies to relational data
3. Further work and references

Main publications

The results presented in Part 4 of the course have been published in the following papers:

- 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]
- 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: [Acciarri *et al.*, 2005; Poggi *et al.*, 2008b; Rodríguez-Muro and Calvanese, 2008; Rodriguez-Muro and Calvanese, 2012]
- Case studies: [Keet *et al.*, 2008; Amoroso *et al.*, 2008; Savo *et al.*, 2010]

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

Query rewriting for more expressive ontology languages

The result presented in Part 4 of the course have recently 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.*, 2009].
- Query rewriting techniques for database inspired constraint languages [Calì *et al.*, 2009a; Calì *et al.*, 2009b].

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**. Preliminary results are in [Calvanese *et al.*, 2007d]. Ongoing work 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 the QUEST 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.

References I

[Acciarri *et al.*, 2005] Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati.

QuOnto: QUerying ONTOlogies.

In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 1670–1671, 2005.

[Amoroso *et al.*, 2008] Alfonso Amoroso, Gennaro Esposito, Domenico Lembo, Paolo Urbano, and Raffaele Vertucci.

Ontology-based data integration with Mastro-i for configuration and data management at SELEX Sistemi Integrati.

In *Proc. of the 16th Ital. Conf. on Database Systems (SEBD 2008)*, pages 81–92, 2008.

[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 2007)*, volume 4801 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2007.

References II

[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 2008)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1440–1457. Springer, 2008.

[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 2009)*, pages 14–30, 2009.

References III

[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 2009)*, pages 77–86, 2009.

[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.

[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 2005)*, pages 602–607, 2005.

References IV

[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 *DL-Lite*$_A$.
In *Proc. of the 2nd Int. Workshop on OWL: Experiences and Directions (OWLED 2006)*, 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 2006)*, pages 260–270, 2006.

[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 2007)*, volume 258 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2007.

References V

[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 2007)*, 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.

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

unibz.it

References VI

[Calvanese *et al.*, 2008a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi.
Data integration through $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ 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 2008)*, 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, *Semantic Technologies for Informations Systems – 5th Int. Reasoning Web Summer School (RW 2009)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.

References VII

[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 2010)*, volume 619 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 3.1–3.12, 2010.

[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 2008)*, volume 353 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2008.

References VIII

[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.

[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 2008)*, volume 432 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2008.

[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 2008)*, pages 285–295, 2008.

References IX

[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 2009)*, pages 836–840, 2009.

[Pérez-Urbina *et al.*, 2009] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks.
A comparison of query rewriting techniques for *DL-lite*.
In *Proc. of the 22nd Int. Workshop on Description Logic (DL 2009)*, volume 477 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2009.

[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 2008 DC)*, 2008.

References X

[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 2008)*, 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 2012)*, 2012.

[Rosati, 2008] Riccardo Rosati.
Finite model reasoning in *DL-Lite*.
In *Proc. of the 5th European Semantic Web Conf. (ESWC 2008)*, 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 2010)*, volume 573 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 20–31, 2010.

References XI

[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 2010)*, volume 573 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 102–113, 2010.