Towards Systems for Ontology-based Data Access and Integration using Relational Technology

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma, Italy

University of Toronto – October 5, 2010

Outline

Outline

Semantic Data Access and Integration: a challenge in IT

- Information systems of organizations are typically constituted by several, distributed, heterogeneous data sources: $\Rightarrow$ integrating such information is one of the major challenge in IT
- From [Bernstein & Haas, CACM Sept. 2008]:
  - Large enterprises spend a great deal of time and money on information integration (e.g., 40% of information-technology shops' budget).
  - Market for data integration software estimated to grow from \$2.5 billion in 2007 to \$3.8 billion in 2012 (+8.7% per year) [IDC. Worldwide Data Integration and Access Software 2008-2012 Forecast. Doc No. 211636 (Apr. 2008)]
- Integration is mainly done by humans: current automated tools are largely unsatisfactory.

Semantic Data Access and Integration: a challenge in IT

Desiderata: achieve logical transparency in access to data:

- Hide to the user where and how data are stored.
- Present to the user a conceptual view of the data.
- Use a semantically rich formalism for the conceptual view.

Ontologies can play a key role!

Ontologies

Definition

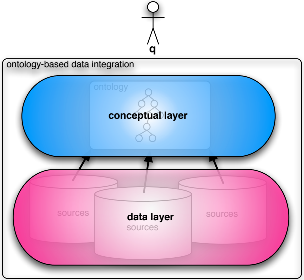
An **ontology** is a representation scheme that describes a **formal conceptualization** of a domain of interest.

The specification of an ontology comprises several levels, and in particular:

- **Intensional level**: specifies a set of conceptual elements and of rules to describe the conceptual structures of the domain.
- **Extensional level**: specifies a set of instances of the conceptual elements described at the intensional level.

Ontology-based data access: conceptual layer & data layer

Ontology-based data access is based on the idea of decoupling information access from data storage.



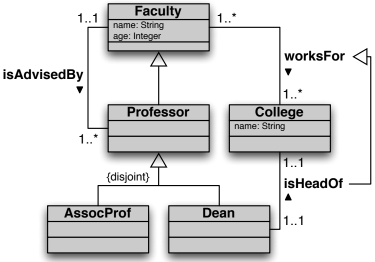
Clients access only the **conceptual layer** ... while the **data layer**, hidden to clients, manages the data.

Intensional level of an ontology language

Ontology languages for the intensional level:
Usually include

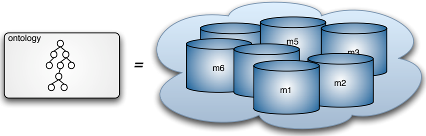
- **Concepts/Classes**
  e.g., Professor, College
- **Properties of concepts**
  e.g., name, age
- **Relationships between concepts**
  e.g., worksFor
- **Properties of relationships**
  e.g., since
- **Constraints**
  e.g., Dean $\sqsubseteq$ Professor

Often are **rendered as a diagram**
e.g., Semantic Network (AI), Entity-Relationship schema (DB), UML Class Diagram (SE)



Ontologies and Reasoning

- Formally we can see ontologies are **logical theories**, and several interpretations may exist that satisfy them (*incomplete information*)



- Reasoning over ontologies amounts to make logical **inference** over them
  - Intensional reasoning: concept/relationship satisfiability, concept/relationship subsumption, etc.
  - Ontology reasoning: ontology satisfiability, instance checking, query answering.

Description Logics are logics specifically designed to represent and reason on structured knowledge:

The domain is composed of objects and is structured into:

- concepts, which correspond to classes, and denote sets of objects
- roles, which correspond to (binary) relationships, and denote binary relations on objects

The knowledge is asserted through so-called assertions, i.e., logical axioms.

Notice these are exactly the constructs at the base of (the intentional level of) ontologies!

One slide (very partial) history of DLs

70's Semantic Networks, Frame Systems:
[Woods75] "**What is a link?**": no clear semantics, reasoning not well understood

80's Description **Logics**, Concept Languages, Terminological Languages.
[BrachmanLevesque84]: "**expressiveness/complexity tradeoff**"
[Patel-Schneider89]: "**Classic**"

90's Focus on assertions (TBox):

[Lenzerini89], : Description logic as formalisation of **conceptual models**: But we need of inverse roles and cardinality restrictions! Also Alex Borgida DLs+DBs!

[Baader90]: **Tableaux** for $\mathcal{ALC}$ with assertions – EXPTIME-completeness

[Schild91], [DeGiacomo95]: **Description logic = Modal Logics for actions** (fancy ones: with inverses, graded modalities, nominals). $\Longrightarrow$ "expressiveness/complexity tradeoff" flatten to EXPTIME-completeness (except for nominals and inverses).
Interestingly, the correspondence already came out in the '80 in discussions between Hector Levesque and Jeff Rosenschein, and as a NP-hardness (in fact EXPTIME-hardness) argument for certain description languages, but was never published and in fact forgotten by the community.

[Horrocks96]: **Optimized tableaux** for expressive DLs as $\mathcal{ALCQI}$, later $\mathcal{SHIQ}$

[CalvaneseLenzeriniDeGiacomo98] **Conjunctive Queries** on DLs are decidable!

2000 Semantic Web: **OWL-DL W3C Standard**!!! Horrocks and Patel-Schneider manage to stick to scientific grounds in defining the standard!!!

Current New focus on tractability:

- Dresden: $\mathcal{EL}$
- Rome: **DL-Lite**.

Current applications of Description Logics

DLs have evolved from being used "just" in KR.

Novel applications of DLs:

- Databases:
  - schema design, schema evolution
  - query optimization
  - integration of heterogeneous data sources, data warehousing
- **Conceptual modeling**
- Foundation for the Semantic Web (variants of OWL correspond to specific DLs)
- ...

A **Description Logic** is characterized by:

1. A **description language**: how to form concepts and roles
   $\mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{hasChild} \sqcap \forall \mathsf{hasChild}.(\mathsf{Doctor} \sqcup \mathsf{Lawyer})$

2. A mechanism to assert **intensional knowledge** about concepts and roles (TBox)
   $$\begin{aligned}\mathcal{T} \;=\; \{\; & \mathsf{Father} \;\equiv\; \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{hasChild}, \\ & \mathsf{HappyFather} \;\sqsubseteq\; \mathsf{Father} \sqcap \forall \mathsf{hasChild}.(\mathsf{Doctor} \sqcup \mathsf{Lawyer}) \;\}\end{aligned}$$

3. A mechanism to assert **extensional knowledge** about objects (ABox)
   $$\mathcal{A} \;=\; \{\; \mathsf{HappyFather}(\mathtt{john}), \quad \mathsf{hasChild}(\mathtt{john}, \mathtt{mary}) \;\}$$

4. A set of **inference services**: how to reason on a given KB
   $$\mathcal{T} \;\models\; \mathsf{HappyFather} \sqsubseteq \exists \mathsf{hasChild}.(\mathsf{Doctor} \sqcup \mathsf{Lawyer})$$
   $$\mathcal{T} \cup \mathcal{A} \;\models\; (\mathsf{Doctor} \sqcup \mathsf{Lawyer})(\mathtt{mary})$$

Ontologies and data

- The best current ontology reasoning systems can deal with a moderately large instance level. $\leadsto 10^4$ individuals *(and this is a big achievement of the last years)!*
- But data of interests in typical information systems (and in data integration) are much **larger**
  $\leadsto 10^6 - 10^9$ individuals
- The best technology to deal with large amounts of data are **relational databases**.

Question:

How can we use ontologies together with large amounts of data?

Challenges when integrating data into ontologies

Deal with well-known tradeoff between expressive power of the ontology language and complexity of dealing with (i.e., performing inference over) ontologies in that language.

Requirements come from the specific setting:

- We have to fully take into account the ontology.
  $\leadsto$ **inference**
- We have to deal very large amounts of data.
  $\leadsto$ **relational databases**
- We want flexibility in querying the data.
  $\leadsto$ **expressive query language**
- We want to keep the data in the sources, and not move it around.
  $\leadsto$ **map** data sources to the ontology (Virtual Data Integration)

Questions to be addressed

1. Which is the "right" **ontology language**?
2. Which is the "right" **query language**?
3. How can we bridge the **semantic mismatch** between the ontology and the data sources?
4. How can **tools for ontology-based data access and integration** fully take into account all these issues?

Outline

1. Introduction
2. Querying data through ontologies
3. *DL-Lite*$_{\mathcal{A}}$: an ontology language for accessing data
4. Ontology-based data integration
5. Discussion

Ontology languages vs. query languages

Which query language to use?

Two extreme cases:

1. Just classes and properties of the ontology $\rightsquigarrow$ instance checking
   - Ontology languages are tailored for capturing intensional relationships.
   - They are quite **poor as query languages**:
     Cannot refer to same object via multiple navigation paths in the ontology, i.e., allow only for a limited form of JOIN, namely chaining.

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

A good compromise are (unions of) **conjunctive queries.**

Conjunctive queries (CQs)

A **conjunctive query (CQ)** is a first-order query of the form

$$q(\vec{x}) \leftarrow \exists \vec{y}. R_1(\vec{x}, \vec{y}) \wedge \cdots \wedge R_k(\vec{x}, \vec{y})$$

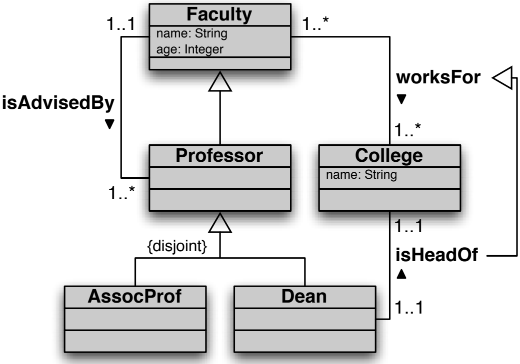
where each $R_i(\vec{x}, \vec{y})$ is an atom using (some of) the free variables $\vec{x}$, the existentially quantified variables $\vec{y}$, and possibly constants.

Note:

- CQs contain no disjunction, no negation, no universal quantification.
- Correspond to SQL/relational algebra select-project-join (SPJ) queries – the most frequently asked queries.
- They also form the core of SPARQL.

Example of conjunctive query

$$
\begin{array}{rcl}
\text{Professor} & \sqsubseteq & \text{Faculty} \\
\text{AssocProf} & \sqsubseteq & \text{Professor} \\
\text{Dean} & \sqsubseteq & \text{Professor} \\
\text{AssocProf} & \sqsubseteq & \neg\text{Dean} \\
\text{Faculty} & \sqsubseteq & \exists\text{age} \\
\exists\text{age}^- & \sqsubseteq & \text{Integer} \\
\exists\text{worksFor} & \sqsubseteq & \text{Faculty} \\
\exists\text{worksFor}^- & \sqsubseteq & \text{College} \\
\text{Faculty} & \sqsubseteq & \exists\text{worksFor} \\
\text{College} & \sqsubseteq & \exists\text{worksFor}^- \\
 & \vdots &
\end{array}
$$



$$
\begin{aligned}
q(nf, nd, av) \;\leftarrow\; & \exists f, c, d. \\
& \text{worksFor}(f, c) \wedge \text{isHeadOf}(d, c) \wedge \text{name}(f, nf) \wedge \text{name}(d, nd) \wedge \\
& \text{age}(f, av) \wedge \text{age}(d, av)
\end{aligned}
$$

Conjunctive queries and SQL – Example

Relational alphabet:

worksFor(fac, coll), isHeadOf(dean, coll), name(p, n), age(p, a)

Query: return name, age, and name of dean of all faculty that have the same age as their dean.

Expressed in SQL:

```
SELECT NF.name, AF.age, ND.name
FROM worksFor W, isHeadOf H, name NF, name ND, age AF, age AD
WHERE W.fac = NF.p   AND  W.fac = AF.p  AND
      H.dean = ND.p  AND  H.dean = AD.p  AND
      W.coll = H.coll  AND  AF.a = AD.a
```

Expressed as a CQ:

$$\begin{aligned} q(nf, af, nd) \leftarrow\ & \mathsf{worksFor}(f1, c1),\ \mathsf{isHeadOf}(d1, c2),\\ & \mathsf{name}(f2, nf),\ \mathsf{name}(d2, nd),\ \mathsf{age}(f3, af),\ \mathsf{age}(d3, ad),\\ & f1 = f2,\ f1 = f3,\ d1 = d2,\ d1 = d3,\ c1 = c2,\ af = ad \end{aligned}$$

Query answering under different assumptions

There are fundamentally different assumptions when addressing query answering in different settings:

- **traditional database assumption**
- **knowledge representation assumption**

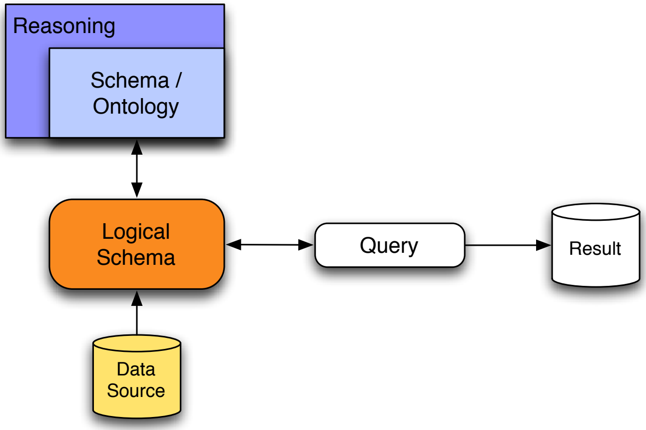
Note: for the moment we assume to deal with an ordinary ABox, which however may be very large and thus is stored in a database.

Query answering under the database assumption

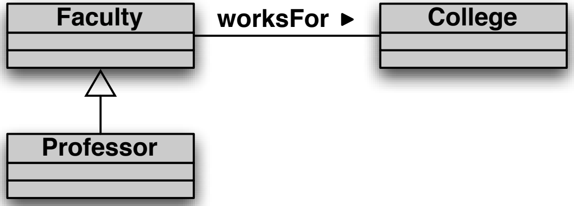
- Data are completely specified (CWA), and typically large.
- Schema/intensional information used in the design phase.
- At **runtime**, the data is assumed to satisfy the schema, and therefore the **schema is not used**.
- Queries allow for complex navigation paths in the data (cf. SQL).

$\leadsto$ Query answering amounts to **query evaluation**, which is computationally easy.

Query answering under the database assumption (cont'd)



Query answering under the database assumption – Example



For each class/property we have a (complete) table in the database.

DB: Faculty = { john, mary, paul }
Professor = { john, paul }
College = { collA, collB }
worksFor = { (john,collA), (mary,collB) }

Query: $q(x) \leftarrow \exists c.\, \mathsf{Professor}(x), \mathsf{College}(c), \mathsf{worksFor}(x, c)$

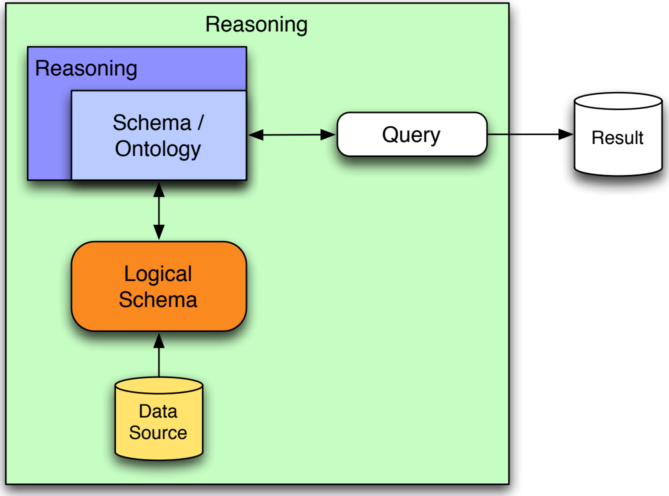
Answer: { john }

Query answering under the KR assumption

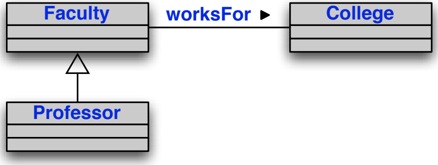
- An ontology imposes constraints on the data.
- Actual data may be incomplete or inconsistent w.r.t. such constraints.
- The system has to take into account the constraints during query answering, and overcome incompleteness or inconsistency.

$\leadsto$ Query answering amounts to **logical inference**, which is computationally more costly.

Query answering under the KR assumption (cont'd)



Query answering under the KR assumption – Example



The tables in the database may be **incompletely specified**, or even missing for some classes/properties.

DB: Professor $\supseteq$ { `john`, `paul` }
College $\supseteq$ { `collA`, `collB` }
worksFor $\supseteq$ { (`john`,`collA`), (`mary`,`collB`) }

Query: $q(x) \leftarrow \text{Faculty}(x)$

Answer: { `john`, `paul`, `mary` }

Certain answers to a query

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology, $\mathcal{I}$ an interpretation for $\mathcal{O}$, and $q(\vec{x}) \leftarrow \exists \vec{y}.\, conj(\vec{x}, \vec{y})$ a CQ.

Def.: The **answer** to $q(\vec{x})$ over $\mathcal{I}$, denoted $q^{\mathcal{I}}$

... is the set of **tuples** $\vec{c}$ **of constants of** $\mathcal{A}$ such that the formula $\exists \vec{y}.\, conj(\vec{c}, \vec{y})$ evaluates to true in $\mathcal{I}$.

We are interested in finding those answers that hold in all models of an ontology.

Def.: The **certain answers** to $q(\vec{x})$ over $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, denoted $cert(q, \mathcal{O})$

... are the **tuples** $\vec{c}$ **of constants of** $\mathcal{A}$ such that $\vec{c} \in q^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{O}$.

Data complexity

Various parameters affect the complexity of query answering over an ontology.

Depending on which parameters we consider, we get different complexity measures:

- **Data complexity**: only the size of the ABox (i.e., the data) matters.
  TBox and query are considered fixed.
- Schema complexity: only the size of the TBox (i.e., the schema) matters.
  ABox and query are considered fixed.
- Combined complexity: no parameter is considered fixed.

In the integration setting, **the size of the data largely dominates** the size of the conceptual layer (and of the query).

$\leadsto$ **Data complexity** is the relevant complexity measure.

Complexity of query answering in ontologies

Studied extensively for (unions of) CQs and various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in LogSpace [(2)]
OWL 2 (and less)	2ExpTime-complete	coNP-hard[(1)]

[(1)] Already for a TBox with a single disjunction!. [(2)] This is what we need!

Question

- Can we find interesting DLs for which the query answering problem can be solved efficiently (i.e., in LogSpace)?
- Can we leverage relational database technology for query answering?

Answer

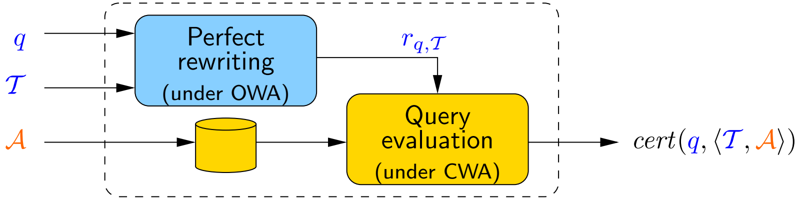
Yes, but we need new foundations!
No more tableaux coming from logic, but **chase** coming from databases as main took for reasoning!

Inference in query answering

To be able to deal with data efficiently, we need to separate the contribution of $\mathcal{A}$ from the contribution of q and $\mathcal{T}$.

$\rightsquigarrow$ Query answering by **query rewriting**.

Query rewriting



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

1. Perfect rewriting: generate a new query $r_{q,\mathcal{T}}$ from q and $\mathcal{T}$.
2. Query evaluation: evaluate $r_{q,\mathcal{T}}$ over the ABox $\mathcal{A}$ seen as a complete database.
   $\leadsto$ Produces $cert(q, \langle\mathcal{T}, \mathcal{A}\rangle)$.

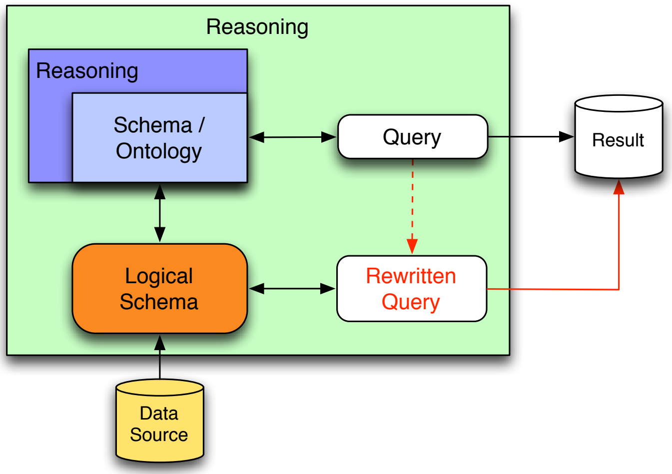
Note: The "always" holds if we pose no restriction on the language in which to express the rewriting $r_{q,\mathcal{T}}$.

Language of the rewriting

The expressiveness of the ontology language affects the **query language into which we are able to rewrite CQs**:

- When we can rewrite into FOL/SQL.
  $\leadsto$ Query evaluation can be done in SQL, i.e., via an RDBMS (*Note:* FOL is in LogSpace).
- When we can rewrite into an NLogSpace-hard language.
  $\leadsto$ Query evaluation requires (at least) linear recursion.
- When we can rewrite into a PTime-hard language.
  $\leadsto$ Query evaluation requires full recursion (e.g., Datalog).
- When we can rewrite into a coNP-hard language.
  $\leadsto$ Query evaluation requires (at least) power of Disjunctive Datalog.

Query rewriting (cont'd)



Outline

The *DL-Lite* Family

The **DL-Lite family** is a family of DL carefully designed to provide robust foundations for Ontology-Based Data Access: Query answering for UCQ is:

- NP-complete in query complexity – as relational DBs
- $\mathrm{PT{\small IME}}$ in the size of the TBox
- $\mathrm{L{\small OG}S{\small PACE}}$ in size of ABox (data complexity) – as relational DBs
- queries can be rewritten into FOL/SQL – allows delegating reasoning on data to a RDMBS!

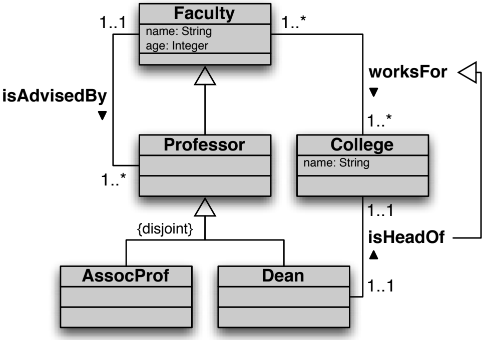
Inference based on (inverted) chase and not on tableaux!

Here we consider **DL-Lite**$_{\mathcal{A}}$, which is one of the most powerful *DL-Lite*'s.

ISA between classes	$A_1 \sqsubseteq A_2$
Disjointness between classes	$A_1 \sqsubseteq \neg A_2$
Domain and range of properties	$\exists P \sqsubseteq A_1 \quad \exists P^- \sqsubseteq A_2$
Mandatory participation *(min card = 1)*	$A_1 \sqsubseteq \exists P \quad A_2 \sqsubseteq \exists P^-$
Functionality of relations *(max card = 1)*	$(\textbf{funct } P) \quad (\textbf{funct } P^-)$
ISA between properties	$Q_1 \sqsubseteq Q_2$
Disjointness between properties	$Q_1 \sqsubseteq \neg Q_2$

Note: *DL-Lite*$_{\mathcal{A}}$ can be extended to capture also min cardinality constraints ($A \sqsubseteq \leq nQ$) and max cardinality constraints ($A \sqsubseteq \geq nQ$) (not considered here for simplicity).

Example



$$
\begin{array}{rcl}
\text{Professor} & \sqsubseteq & \text{Faculty} \\
\text{AssocProf} & \sqsubseteq & \text{Professor} \\
\text{Dean} & \sqsubseteq & \text{Professor} \\
\text{AssocProf} & \sqsubseteq & \neg\text{Dean} \\
\\
\text{Faculty} & \sqsubseteq & \exists \mathit{age} \\
\exists \mathit{age}^- & \sqsubseteq & \texttt{xsd:integer} \\
& (\textbf{funct}\ \mathit{age}) & \\
\\
\exists\text{worksFor} & \sqsubseteq & \text{Faculty} \\
\exists\text{worksFor}^- & \sqsubseteq & \text{College} \\
\text{Faculty} & \sqsubseteq & \exists\text{worksFor} \\
\text{College} & \sqsubseteq & \exists\text{worksFor}^- \\
\\
\exists\text{isHeadOf} & \sqsubseteq & \text{Dean} \\
\exists\text{isHeadOf}^- & \sqsubseteq & \text{College} \\
\text{Dean} & \sqsubseteq & \exists\text{isHeadOf} \\
\text{College} & \sqsubseteq & \exists\text{isHeadOf}^- \\
\text{isHeadOf} & \sqsubseteq & \text{worksFor} \\
& (\textbf{funct}\ \text{isHeadOf}) & \\
& (\textbf{funct}\ \text{isHeadOf}^-) & \\
& \vdots &
\end{array}
$$

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

- Essentially, captures all the basic constructs of UML Class Diagrams and of the ER Model . . .
- . . . **except covering constraints** in generalizations. – if we add them, query answering becomes coNP-hard in data complexity
- A substantial fragment of it, chosen as one one of the three standard OWL 2 Profiles: OWL 2 QL.
- Extends (the DL compatible part of) the ontology language RDFS.
- Completely symmetric w.r.t. direct and inverse properties. roles are always navigable in the two directions
- Non trivial, e.g., does not enjoy the finite model property, i.e., reasoning and query answering differ depending on whether we consider or not also infinite models.

DL-Lite$_{\mathcal{A}}$ does not have the finite model property

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

Example

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

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

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.

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

TBox assertions:

- Class (concept) inclusion assertions: $B \sqsubseteq C$, with:

$$\begin{array}{rcl} B & \longrightarrow & A \mid \exists Q \\ C & \longrightarrow & B \mid \neg B \end{array}$$

- Property (role) inclusion assertions: $Q \sqsubseteq R$, with:

$$\begin{array}{rcl} Q & \longrightarrow & P \mid P^- \\ R & \longrightarrow & Q \mid \neg Q \end{array}$$

- Functionality assertions: $(\textbf{funct}\ Q)$
- **Proviso:** functional properties cannot be specialized.

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

Note: *DL-Lite*$_{\mathcal{A}}$ distinguishes also between object and data properties (ignored here).

SAPIENZA
UNIVERSITÀ DI ROMA

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

Construct	Syntax	Example	Semantics
atomic conc.	A	Doctor	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
exist. restr.	$\exists Q$	∃child$^-$	$\{d \mid \exists e. (d,e) \in Q^{\mathcal{I}}\}$
at. conc. neg.	$\neg A$	¬Doctor	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
conc. neg.	$\neg\exists Q$	¬∃child	$\Delta^{\mathcal{I}} \setminus (\exists Q)^{\mathcal{I}}$
atomic role	P	child	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse role	P^-	child$^-$	$\{(o,o') \mid (o',o) \in P^{\mathcal{I}}\}$
role negation	$\neg Q$	¬manages	$(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$
conc. incl.	$B \sqsubseteq C$	Father ⊑ ∃child	$B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
role incl.	$Q \sqsubseteq R$	hasFather ⊑ child$^-$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
funct. asser.	$(\textbf{funct}\ Q)$	(**funct** succ)	$\forall d,e,e'.(d,e) \in Q^{\mathcal{I}} \wedge (d,e') \in Q^{\mathcal{I}} \rightarrow e = e'$
mem. asser.	$A(c)$	Father($\mathtt{bob}$)	$c^{\mathcal{I}} \in A^{\mathcal{I}}$
mem. asser.	$P(c_1, c_2)$	child($\mathtt{bob}, \mathtt{ann}$)	$(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in P^{\mathcal{I}}$

DL-Lite$_{\mathcal{A}}$ (as all DLs of the *DL-Lite* family) adopts the Unique Name Assumption (UNA), i.e., different individuals denote different objects.

Query answering in *DL-Lite*$_{\mathcal{A}}$

- We study answering of UCQs over *DL-Lite*$_{\mathcal{A}}$ ontologies via query rewriting.
- We first consider query answering over **satisfiable ontologies**, i.e., that admit at least one model.
- Then, we show how to exploit query answering over satisfiable ontologies to establish ontology satisfiability.

Remark

we call **positive inclusions (PIs)** assertions of the form

$$\begin{array}{lcl} B_1 & \sqsubseteq & B_2 \\ Q_1 & \sqsubseteq & Q_2 \end{array}$$

whereas we call negative inclusions (NIs) assertions of the form

$$\begin{array}{lcl} B_1 & \sqsubseteq & \neg B_2 \\ Q_1 & \sqsubseteq & \neg Q_2 \end{array}$$

Query answering over satisfiable *DL-Lite*$_\mathcal{A}$ ontologies

Theorem

Let q be a boolean UCQs and $\mathcal{T} = \mathcal{T}_{\mathrm{PI}} \cup \mathcal{T}_{\mathrm{NI}} \cup \mathcal{T}_{\mathsf{funct}}$ be a TBox s.t.

- $\mathcal{T}_{\mathrm{PI}}$ is a set of PIs
- $\mathcal{T}_{\mathrm{NI}}$ is a set of NIs
- $\mathcal{T}_{\mathsf{funct}}$ is a set of functionalities.

For each ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ **is satisfiable**, we have that

$$\langle \mathcal{T}, \mathcal{A} \rangle \models q \textbf{ iff } \langle \mathcal{T}_{\mathbf{PI}}, \mathcal{A} \rangle \models q.$$

Proof [intuition]

q is a positive query, i.e., it does not contain atoms with negation nor inequality. $\mathcal{T}_{\mathrm{NI}}$ and $\mathcal{T}_{\mathsf{funct}}$ only contribute to infer new negative consequences, i.e, sentences involving negation.

If q is non-boolean, we have that $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = cert(q, \langle \mathcal{T}_{\mathrm{PI}}, \mathcal{A} \rangle)$.

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

Theorem (Separability)

Satisfiability of a *DL-Lite*$_{\mathcal{A}}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ can be reduced to evaluation of a first order query over $\mathcal{A}$, obtained by the union of

(a) FOL queries expressing the violation of the functionalities in $\mathcal{T}$ and

(b) UCQs produced by the query rewriting procedure (which depends only on the PIs in $\mathcal{T}$) applied to the CQ expressing the violation of the NIs in $\mathcal{T}$.

Note that satisfiability in DL-Lite$_{\mathcal{A}}$ *can be done in* LogSpace *w.r.t. the data, using RDMBS technology.*

Query answering in *DL-Lite*$_{\mathcal{A}}$

Query rewriting

To compute the perfect rewriting, starting from the original (U)CQ, iteratively get a CQ to be processed and either:

- **Expand** positive inclusions & simplify redundant atoms, or
- **Unify** atoms in the CQ to obtain a more specific CQ to be further expanded.

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

Query answering

Based on query rewriting: given an (U)CQ and an ontology:

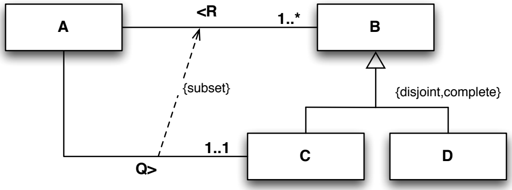
1. **Compute its perfect rewriting**, which is a UCQ;
2. **Evaluate the perfect rewriting** on the ABox seen as a DB.

Recall: negative inclusions and functionalities play a role in ontology satisfiability, but not in query answering.

Example

Consider the *DL-Lite*$_{\mathcal{A}}$ **TBox** $\mathcal{T}$:

$\exists R \sqsubseteq B \quad \exists R^- \sqsubseteq A$
$A \sqsubseteq \exists R^-$
$\exists Q \sqsubseteq A \quad \exists Q^- \sqsubseteq C$
$A \sqsubseteq \exists Q \quad (\textbf{funct } Q)$
$C \sqsubseteq B \quad D \sqsubseteq B$
$C \sqsubseteq \neg D$
$B \sqsubseteq C \sqcup D$ *not expressible!*
$Q \sqsubseteq R^-$



and the **ABox:**

$$\mathcal{A} = \{A(a)\}$$

Compute the answer to the **queries**:

$$\begin{array}{rcl} q(x) & \leftarrow & Q(x,y), R(y,z). \\ q'() & \leftarrow & B(x). \end{array}$$

Example (solution)

Rewritings:

$$
\begin{array}{rcll}
q(x) & \leftarrow & Q(x,y), R(y,z). & \\
q(x) & \leftarrow & Q(x,y), Q(z,y). & Q \sqsubseteq R^- \\
q(x) & \leftarrow & Q(x,y). & \text{unify: } z = x \\
q(x) & \leftarrow & A(x). & A \sqsubseteq \exists Q \\
 & & & \Longrightarrow \text{ answer } x = a
\end{array}
$$

$$
\begin{array}{rcll}
q'() & \leftarrow & B(x). & \\
q'() & \leftarrow & R(x,y). & \exists R \sqsubseteq B \\
q'() & \leftarrow & A(y). & A \sqsubseteq \exists R^- \\
 & & & \Longrightarrow \text{ answer } \mathit{true} \text{ (by } y = a)
\end{array}
$$

Ontology satisfiability and all classical DL reasoning tasks are:

- Efficiently tractable in the size of TBox (i.e., PTIME).
- Very efficiently tractable in the size of the ABox (i.e., LOGSPACE).

In fact, reasoning can be done by constructing suitable FOL/SQL queries and evaluating them over the ABox (**FOL-rewritability**).

Query answering for CQs and UCQs is:

- PTIME in the size of TBox.
- LOGSPACE 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.

Can we go beyond *DL-Lite*$_\mathcal{A}$?

By adding essentially any other DL construct, e.g., union ($\sqcup$), value restriction ($\forall R.C$), etc., without some limitations we lose these nice computational properties (see later).

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

	lhs	rhs	funct.	Prop. incl.	Data complexity of query answering
0	$DL\text{-}Lite_{\mathcal{A}}$		$\surd^*$	$\surd^*$	in LogSpace
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**

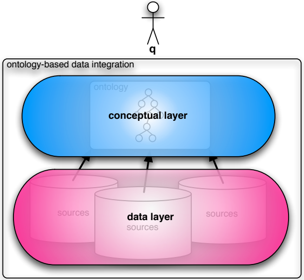
Notes:

- * 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! $\rightsquigarrow$ **No** hope of including **covering constraints**.

Outline

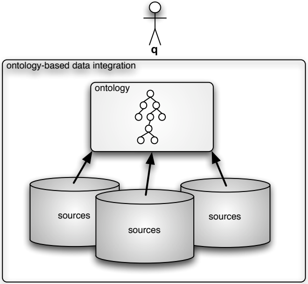
Ontology-based data integration: conceptual layer & data layer

Ontology-based data integration is based on the idea of decoupling information access from data storage.



Clients access only the **conceptual layer** ... while the **data layer**, hidden to clients, manages the data.
$\leadsto$ *Technological concerns (and changes) on the managed data become fully transparent to the clients.*

Ontology-based data integration: architecture

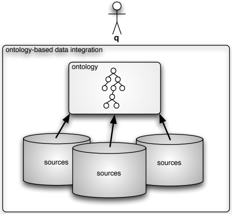


Based on three main components:

- **Ontology**, used as the conceptual layer to give clients a unified conceptual "global view" of the data.
- **Data sources**, these are external, independent, heterogeneous, multiple information systems.
- **Mappings**, which semantically link data at the sources with the ontology *(key issue!)*

Ontology-based data integration: the conceptual layer

The ontology is used as the conceptual layer, to give clients a unified conceptual global view of the data.

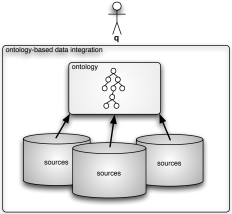


Note: in standard information systems, UML Class Diagram or ER is used at **design time**, ...

... here we use ontologies at **runtime**!

Ontology-based data integration: the sources

Data sources are external, independent, heterogeneous, multiple information systems.

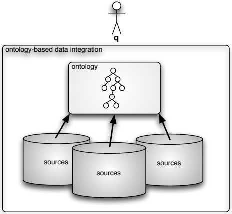


By now we have industrial solutions for:

- Distributed database systems & Distributed query optimization
- Tools for source wrapping
- Systems for database federation, e.g., IBM Information Integrator

Ontology-based data integration: the sources

Data sources are external, independent, heterogeneous, multiple information systems.



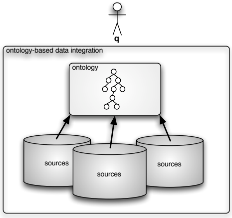
Based on these industrial solutions we can:

1. Wrap the sources and see all of them as relational databases.
2. Use federated database tools to see the multiple sources as a single one.

$\leadsto$ We can see the sources as a single (remote) relational database.

Ontology-based data integration: mappings

Mappings semantically link data at the sources with the ontology.



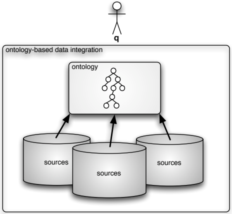
Scientific literature on data integration in databases has shown that ...

... generally we cannot simply **map** single relations to single elements of the global view (the ontology) ...

... we need to rely on **queries**!

Ontology-based data integration: mappings

Mappings semantically link data at the sources with the ontology.

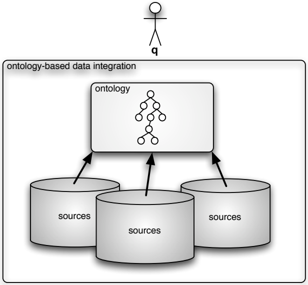


Several general forms of mappings based on queries have been considered:

- GAV: map a query over the source to an element in the global view
  – most used form of mappings
- LAV: map a relation in the source to a query over the global view
  – mathematically elegant, but difficult to use in practice (data in the sources are not clean enough!)
- GLAV: map a query over the sources to a query over the global view
  – the most general form of mappings

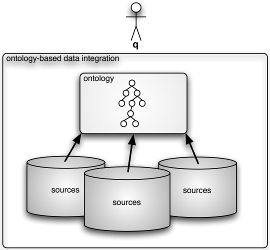
This is a key issue (more on this later).

Ontology-based data integration: the *DL-Lite* solution



- We require the data sources to be **wrapped** and presented as relational sources. $\rightsquigarrow$ *"standard technology"*
- We make use of a **data federation tool**, such as IBM Information Integrator, to present the yet to be (semantically) integrated sources as a single relational database. $\rightsquigarrow$ *"standard technology"*
- We make use of the **DL-Lite** technology presented above for the conceptual view on the data, to **exploit effectiveness of query answering**. $\rightsquigarrow$ *"new technology"*

Ontology-based data integration: the *DL-Lite* solution



Are we done? Not yet!

- The (federated) source database is **external** and **independent** from the conceptual view (the ontology).
- **Mappings** relate information in the sources to the ontology. $\rightsquigarrow$ sort of virtual ABox

  We use GAV (global-as-view) mappings: the result of an (arbitrary) SQL query on the source database is considered a (partial) extension of a concept/role.
- Moreover, we properly deal with the notorious **impedance mismatch problem**!

Impedance mismatch problem

The impedance mismatch problem

- In **relational databases**, information is represented in forms of tuples of **values**.
- In **ontologies** (or more generally object-oriented systems or conceptual models), information is represented using both **objects** and values ...
  - ... with objects playing the main role, ...
  - ... and values a subsidiary role as fillers of object's attributes.

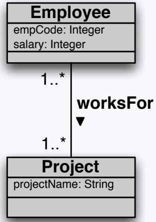
$\leadsto$ *How do we reconcile these views?*

Solution: We need **constructors** to create objects of the ontology out of tuples of values in the database.

Note: from a formal point of view, such constructors can be simply Skolem functions!

Ontology with mappings – Example

TBox $\mathcal{T}$ (UML)



federated schema of the DB $\mathcal{S}$

$\mathrm{D}_1[\mathit{SSN}\colon \mathrm{String}, \mathit{PrName}\colon \mathrm{String}]$
Employees and Projects they work for

$\mathrm{D}_2[\mathit{Code}\colon \mathrm{String}, \mathit{Salary}\colon \mathrm{Int}]$
Employee's Code with salary

$\mathrm{D}_3[\mathit{Code}\colon \mathrm{String}, \mathit{SSN}\colon \mathrm{String}]$
Employee's Code with SSN

...

Mapping $\mathcal{M}$

M_1: `SELECT SSN, PrName FROM` D_1 $\leadsto$ Employee(**pers**(*SSN*)), Project(**proj**(*PrName*)), projectName(**proj**(*PrName*), *PrName*), workFor(**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*)

DL-Lite$_{\mathcal{A}}$ query answering for data integration

Given a (U)CQ q and $\mathcal{O}_m = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ (assumed satisfiable, i.e., there exists at least one model for $\mathcal{O}_m$), we compute $cert(q, \mathcal{O}_m)$ as follows:

1. Using $\mathcal{T}$, **reformulate** CQ q as a union $r_{q,\mathcal{T}}$ of CQs.
2. Using $\mathcal{M}$, **unfold** $r_{q,\mathcal{T}}$ to obtain a union $unfold(r_{q,\mathcal{T}})$ of CQs.
3. **Evaluate** $unfold(r_{q,\mathcal{T}})$ directly over $\mathcal{S}$ using RDBMS technology.

Correctness of this algorithm shows FOL-reducibility of query answering.
$\rightsquigarrow$ Query answering can again be done using **RDBMS technology**.

Computational complexity of query answering

Theorem

Query answering in a *DL-Lite*$_\mathcal{A}$ ontology with mappings $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ 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. **LogSpace** in the size of the **database** $\mathcal{S}$, in fact FOL-rewritable.

Can we move to LAV or GLAV mappings?
No, if we want to have DL-Lite$_\mathcal{A}$ *TBoxes and stay in* LOGSPACE*!*

Alternatively, we can have LAV or GLAV mappings, but we have to **renounce to use functionalities** *in the TBox (thus not having DL-Lite*$_\mathcal{A}$ *TBoxes) and* **limit** *the form of the queries in the mapping (essentially CQs over both the sources and the ontology), if we want to stay in* LOGSPACE.

Outline

Beyond union of conjunctive queries

Till now we have assumed that the client queries are UCQs (aka positive queries).

Can we go beyond UCQ? Can we go to full **FOL/SQL queries**?

- No! Answering FOL queries in presence of incomplete information is undecidable: Consider an empty source (no data), still a (boolean) FOL query may return *true* because it is valid! (FOL validity is undecidable)
- Yes! With some compromises:
  Query what the ontology **knows** about the domain, not what is **true** in the domain!
  On knowledge we have complete information, so evaluating FOL queries is LogSpace.

SparSQL

Full **SQL**, but with relations in the FROM clause that are UCQs, expressed in **SPARQL**, over the ontology.

- **SPARQL** queries are used to query what is **true** in the domain.
- **SQL** is used to query what the ontology **knows** about the domain.

Example: negation

Return all known people that are neither known to be male nor known to be female.

```
SELECT persons.x FROM SparqlTable(SELECT ?x
                  WHERE {?x rdf:type 'Person'}
                 ) persons
EXCEPT  ( SELECT males.x FROM
SparqlTable(SELECT ?x
                  WHERE {?x rdf:type 'Male'}
                 ) males
UNION SELECT females.x FROM SparqlTable(SELECT
?x
                  WHERE {?x rdf:type 'Female'}
                 ) females
)
```

Example: aggregates

Return the people and the number of their known spouses, but only if they are known to be married to at least two people.

```
SELECT marriage.x, count(marriage.y) FROM
SparqlTable(SELECT ?x ?y
                 WHERE {?x :MarriedTo ?y}
                ) marriage
GROUP BY marriage.x HAVING count(marriage.y) >= 2
```

SparSQL in *DL-Lite*$_{\mathcal{A}}$

Answering of SparSQL queries in *DL-Lite*$_{\mathcal{A}}$:

1. Expand and unfold the UCQs (in the SparqlTables) as usual in *DL-Lite*$_{\mathcal{A}}$ $\leadsto$ an SQL query over the sources for each SparqlTable in the FROM clauses.
2. Substitute SparqlTables with the new SQL queries. $\leadsto$ the result is again an SQL query over the sources!
3. Evaluate the resulting SQL query over the sources

Note works both for large ABoxes and for data integration!

Data completion approach to *DL-Lite*$_{\mathcal{A}}$

The approach presented is essentially "**hands-off w.r.t. the data**": a key features in several domains including data integration.

But what if we allow **LogSpace/NLogSpace/PTime computation over the data**?

See:

> *The Combined Approach to Query Answering in DL-Lite. By Kontchakov, Lutz,* **Toman**, *Wolter and Zakharyaschev.*
> *KR2010 Ray Reiter Best Paper Award!*

Case studies in industrial settings

We are conducting extensive experimentations with some companies and organizations:

- SELEX, world leading company in the provision of air traffic systems: integration of disperse data about obsolescence of apparatus components (2008)
- Monte Paschi Siena, one of the main Italian banks: pilot project on data concerning grant credit risk estimation (2008); extensive use as support in the re-engineering of the information system after merging with Banca Antonveneta (2010-2012)
- Accenture, a world leading company in ITC consultancy: pilot project on the ADSL traffic domain (2010)
- SAPIENZA, University of Rome: ontology-based data access to the informative system of the university (2009-ongoing)

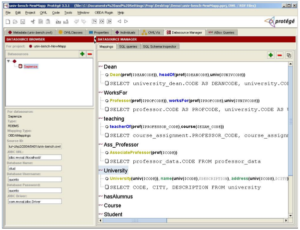
The QuOnto-Mastro tools

- QuOnto is a tool for representing and reasoning over ontologies of the *DL-Lite* family.
- Basic functionalities:
  - Ontology representation and classification
  - Ontology satisfiability check
  - Intensional reasoning services: concept/property subsumption and disjunction, concept/property satisfiability
  - Query Answering of UCQs
- Includes also full support for:
  - Identification path constraints
  - Denial constraints
  - Epistemic queries –*expressed in SparSQL*
  - Epistemic constraints –*expressed as boolean SparSQL queries*
- Reasoning services are highly optimized
- Can be used with internal and external DBMS (include drivers for Oracle, DB2, IBM Information Integrator, SQL Server, MySQL, etc.)
- Implemented in Java

Sapienza Università di Roma

The QuOnto-Mastro tools (cont'd)

- Mastro uses QuOnto at its core and extends its functionalities providing support for specifying and managing mappings between *DL-Lite*$_{\mathcal{A}}$ ontologies and data stored in external systems (e.g., Oracle, DB2, IBM Information Integrator, etc.), and for extracting data from such systems by querying the ontology.
- An open source plugin for Protégé that extends the ontology editor with facilities to design Mappings towards those external DBMS is available.



- The plugin for Protégé 4 can downloaded at `www.dis.uniroma1.it\quonto`.

Wrapping up

- Ontology-based data access and integration is a challenging problem with great practical relevance.
- In this setting, the size of the data is the relevant parameter that must guide technological choices.
- Currently, scalability w.r.t. the size of the data can be achieved only by relying on commercial technologies for managing the data, i.e., relational DBMS systems and federation tools.
- In order to tailor semantic technologies so as to provide a good compromise between expressivity and efficiency, requires a thorough understanding of the semantic and computational properties of the adopted formalisms.
- We have now gained such an understanding, that allows us to develop very good solutions for ontology-based data access and integration.
- One of the three OWL 2 profiles, namely "OWL 2 QL", is directly based on this understanding.

Acknowledgements

People involved in this work:

- Sapienza Università di Roma
  - Giuseppe De Giacomo
  - Claudio Corona
  - Domenico Lembo
  - Maurizio Lenzerini
  - Antonella Poggi
  - Riccardo Rosati
  - Marco Ruzzi
  - Domenico Fabio Savo
- Libera Università di Bolzano
  - Diego Calvanese
  - Mariano Rodriguez Muro
- Students (thanks!)