QuOnto: ontology-based data access and integration using relational technology

Giuseppe De Giacomo

Sapienza Università di Roma

Semantic Days 2009

Motivation: ontologies and data

- The best current standard DL reasoning systems can deal with moderately large ABoxes. $\rightsquigarrow 10^4$ individuals *(and this is a big achievement of the last years)!*
- But data of interests in typical information systems are much **larger** $\rightsquigarrow 10^6 - 10^9$ 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?

Answer:

Yes, by using the system QuOnto.

Which query language?

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

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

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

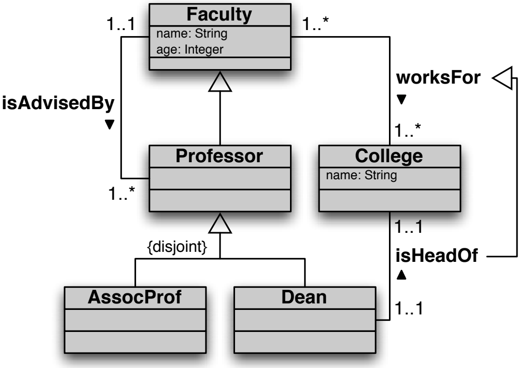
where each $R_i(\vec{x}, \vec{y})$ is an atom using (some of) the free variables $\vec{x}$, the existentially quantified variables $\vec{y}$, and possibly constants.

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 can form the core of SPARQL queries.

Example of conjunctive query

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



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

Query answering: 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 $\mathcal{O}$.

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 to scale with the data.

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

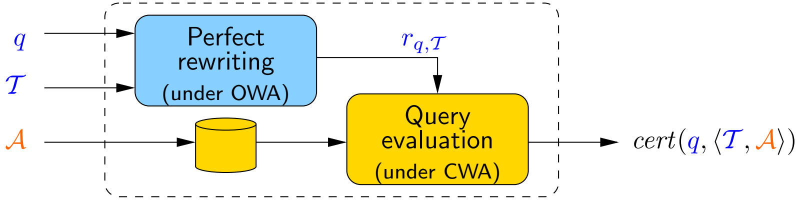
Yes, but we need new foundations!
No more tableaux coming from logic, but **chase** coming from databases as main tool for reasoning!

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 $\mathcal{T}$.

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

Query rewriting



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

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

The QuOnto description logic: $\textit{DL-Lite}_{\mathcal{A}}$

- QuOnto is based on $\textit{DL-Lite}_{\mathcal{A}}$.
- $\textit{DL-Lite}_{\mathcal{A}}$ is carefully designed to provide robust foundations for Ontology-Based Data Access: Query answering for UCQ is:
  - NP-complete in query complexity – as relational DBs
  - PTime in the size of the TBox
  - LogSpace 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.

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

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	$Q_1 \sqsubseteq \neg Q_2$

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

- 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, chosen as one one of the three candidate OWL 2 Profiles: OWL 2 QL.
- Extends (the DL compatible part of) the ontology language RDFS.

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! $\leadsto$ No hope of including covering constraints.

Example

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

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

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

ABox: $\text{teaches}(\texttt{john}, \texttt{kbdb})$
$\text{Professor}(\texttt{mary})$

It is easy to see that $\mathit{Eval}(\mathit{SQL}(r_{q,\mathcal{T}}), \mathit{DB}(\mathcal{A}))$ in this case produces as answer $\{\texttt{john}, \texttt{mary}\}$.

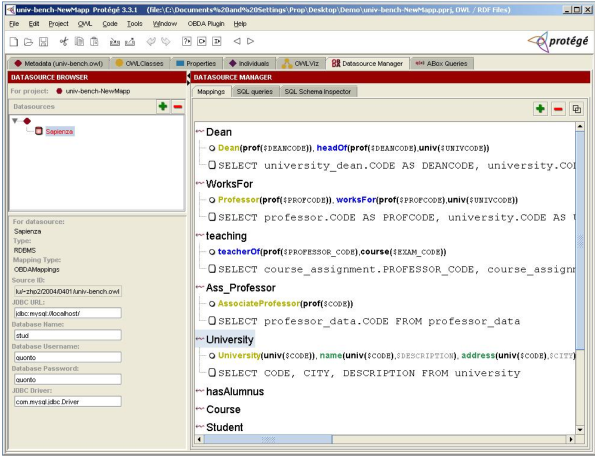
QuOnto as software artifact

- Includes support for:
  - *DL-Lite*$_{\mathcal{A}}$
  - Identification path constrains
  - Denial constrains
  - Epistemic constraints
  - Union of conjunctive queries – *expressed in Datalog or SPARQL*
  - Epistemic queries –*expressed in SparSQL*
- 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 – *API are available for selected projects upon request*
- Several wrapped versions publicly available at:
  `http://www.dis.uniroma1.it/~quonto/` *(or just google "quonto")*

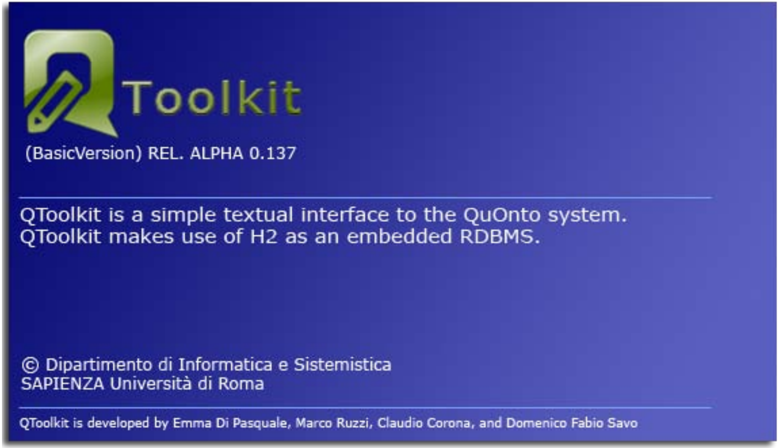
QuOnto wrapped versions
http://www.dis.uniroma1.it/~quonto/

DIG Server wrapper + ODBA Protégé plugin

by Mariano Rodriguez-Muro, Univ. Bolzano



Qtoolkit: simple graphical interface, only standard ABoxes (no connection to external DBs)



QuOnto wrapped versions
http://www.dis.uniroma1.it/~quonto/

ROWLkit: first implementation of the OWL 2 QL Profile

Conclusions

- Ontology-based data access and data integration is ready for prime time – *see Calvanese&De Giacomo's tutorial*
- QuOnto provides serious proof of concept of this.
- We are successfully applying QuOnto in various full-fledged case studies – *see Diego Calvanese's talk*
- We are currently looking for projects where to apply such technology further!
- This technology is ready to become a product!

Acknowledgements

People involved in this work:

- Sapienza Università di Roma
  - Claudio Corona
  - Domenico Lembo
  - Maurizio Lenzerini
  - Antonella Poggi
  - Riccardo Rosati
  - Marco Ruzzi
  - Domenico Fabio Savo
- Libera Università di Bolzano
  - Diego Calvanese
  - Mariano Rodriguez Muro
- Several past master and PhD students (thanks!)