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. $\sim 10^4$ individuals \textit{(and this is a big achievement of the last years)!}

- But data of interests in typical information systems are much larger $\sim 10^6 - 10^9$ 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 to use?

Two extreme cases:

1. **Just classes and properties** of the ontology $\leadsto$ 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**.
A **conjunctive query (CQ)** is a first-order query of the form

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

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

\[ q(nf, nd, av) \leftarrow \exists f, c, d. \]

\[ \text{worksFor}(f, c) \land \text{isHeadOf}(d, c) \land \text{name}(f, nf) \land \text{name}(d, nd) \land \text{age}(f, av) \land \text{age}(d, av) \]
Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology, $\mathcal{I}$ an interpretation for $\mathcal{O}$, and $q(\vec{x}) \leftarrow \exists \vec{y}. \text{conj}(\vec{x}, \vec{y})$ 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 $\exists \vec{y}. \text{conj}(\vec{c}, \vec{y})$ 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 $\text{cert}(q, \mathcal{O})$

... 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:

<table>
<thead>
<tr>
<th></th>
<th>Combined complexity</th>
<th>Data complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain databases</td>
<td>NP-complete</td>
<td>in \text{LOGSPACE} \textsuperscript{(2)}</td>
</tr>
<tr>
<td>OWL 2 (and less)</td>
<td>2\text{EXP}T\text{IME}-complete</td>
<td>\text{coNP}-hard \textsuperscript{(1)}</td>
</tr>
</tbody>
</table>

\(\textsuperscript{(1)}\) Already for a TBox with a single disjunction!
\(\textsuperscript{(2)}\) 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 \text{LOGSPACE})?
- Can we leverage relational database technology for query answering?

Answer

**Yes, but we need new foundations!**

No more tableaux coming from logic, but \textit{chase} coming from databases as main tool for reasoning!
To be able to deal with data efficiently, we need to separate the contribution of $A$ from the contribution of $q$ and $T$.

$\leadsto$ Query answering by **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 \( r_{q,T} \).
2. **Query evaluation**: evaluate \( r_{q,T} \) over the ABox \( \mathcal{A} \) seen as a complete database.
   \( \leadsto \) Produces \( \text{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle) \).

Note: The “always” holds if we pose no restriction on the language in which to express the rewriting \( r_{q,T} \).
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**.
  - 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.
  - Query evaluation requires (at least) linear recursion.

- When we can rewrite into a **PTime-hard** language.
  - Query evaluation requires full recursion (e.g., Datalog).

- When we can rewrite into a **coNP-hard** language.
  - Query evaluation requires (at least) power of Disjunctive Datalog.
The **QuOnto** description logic: $DL\text{-}Lite_{\mathcal{A}}$

- **QuOnto** is based on $DL\text{-}Lite_{\mathcal{A}}$.
- $DL\text{-}Lite_{\mathcal{A}}$ 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
  - $P$\text{Time} in the size of the TBox
  - $L$\text{ogSpace} 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.
<table>
<thead>
<tr>
<th>Property Type</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA between classes</td>
<td>$A_1 \sqsubseteq A_2$</td>
</tr>
<tr>
<td>Disjointness between classes</td>
<td>$A_1 \sqsubseteq \neg A_2$</td>
</tr>
<tr>
<td>Domain and range of properties</td>
<td>$\exists P \sqsubseteq A_1 \quad \exists P^- \sqsubseteq A_2$</td>
</tr>
<tr>
<td>Mandatory participation ($\text{min card} = 1$)</td>
<td>$A_1 \sqsubseteq \exists P \quad A_2 \sqsubseteq \exists P^-$</td>
</tr>
<tr>
<td>Functionality of relations ($\text{max card} = 1$)</td>
<td>($\text{funct } P$) ($\text{funct } P^-$)</td>
</tr>
<tr>
<td>ISA between properties</td>
<td>$Q_1 \sqsubseteq Q_2$</td>
</tr>
<tr>
<td>Disjointness between properties</td>
<td>$Q_1 \sqsubseteq \neg Q_2$</td>
</tr>
</tbody>
</table>
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 \textit{coNP}-hard in data complexity.

A substantial fragment, chosen as one of the three candidate **OWL 2 Profiles**: **OWL 2 QL**.

Extends (the DL compatible part of) the ontology language **RDFS**.
# Beyond $DL-Lite_\mathcal{A}$: results on data complexity

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
<th>Function</th>
<th>Prop. Incl.</th>
<th>Data Complexity of Query Answering</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DL-Lite_\mathcal{A}$</td>
<td>$\exists P. A$</td>
<td>$\sqrt{\ast}$</td>
<td>$\sqrt{\ast}$</td>
<td>in LogSpace</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P. A$</td>
<td>$A$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A$</td>
<td>$A</td>
<td>\forall P. A$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A$</td>
<td>$A</td>
<td>\exists P. A$</td>
<td>$\sqrt{\ast}$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P. A</td>
<td>A_1 \cap A_2$</td>
<td>$A$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A</td>
<td>A_1 \cap A_2$</td>
<td>$A</td>
<td>\forall P. A$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A</td>
<td>A_1 \cap A_2$</td>
<td>$A</td>
<td>\exists P. A$</td>
<td>$\sqrt{\ast}$</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P. A</td>
<td>\exists P^\perp. A$</td>
<td>$A</td>
<td>\exists P$</td>
</tr>
<tr>
<td>$A</td>
<td>\exists P</td>
<td>\exists P^\perp$</td>
<td>$A</td>
<td>\exists P</td>
</tr>
<tr>
<td>$A</td>
<td>\neg A$</td>
<td>$A$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A$</td>
<td>$A</td>
<td>A_1 \sqcup A_2$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$A</td>
<td>\forall P. A$</td>
<td>$A$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

**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! ∼ No hope of including covering constraints.
Example

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

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

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

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

It is easy to see that \( \text{Eval}(\text{SQL}(r_q, T), \text{DB}(A)) \) in this case produces as answer \( \{\text{john, mary}\} \).
QuOnto as software artifact

- Includes support for:
  - $DL$-$Lite_A$
  - 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: ontology-based data access

Semantic Days 2009
QuOnto wrapped versions
http://www.dis.uniroma1.it/~quonto/

DIG Server wrapper + ODBA Protégé plugin

by Mariano Rodriguez-Muro, Univ. Bolzano
QuOnto wrapped versions

http://www.dis.uniroma1.it/~quonto/

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

---

**ROWLKit** is a very simple GUI toolkit for OWL 2 QL ontologies. The main features are:
- language check: parses an OWL file and verifies if such file is expressed in the OWL 2 QL fragment
- intensional reasoning: ontology classification and basic classes (and properties) subsupption and satisfiability
- query answering: evaluates union of conjunctive queries expressed in SPARQL

ROWLKit makes use of H2 as an embedded RDBMS.

© Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma

ROWLKit is developed by Marco Ruzzi, Claudio Corona, and Domenico Fabio Savo
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!
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!)