Query answering and query abstraction through ontologies

Maurizio Lenzerini

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

ER Online Summer Seminars

ERROSS 2020

October 14, 2020
The KRDB Research Centre for Knowledge and Data at the Summer Seminars (EROS-2020), an interdisciplinary seminar on topics such as Ontology, Data Semantics, Artificial Intelligence, with the ultimate goal of contributing to the theory and practice of...
ER Online Summer Seminars: very, very interesting event

eroSS

The KRDB Research Centre for Knowledge and Data at the
Summer Seminars" (EROSS-2020), an interdisciplinary sesuch as Ontology, Data Semantics, Artificial Intelligence, S
ultimate goal of contributing to the theory and practice of

concepts carve reality at its joints
ER Online Summer Seminars: very, very, very interesting event

The KRDB Research Centre for Knowledge and Data at the Summer Seminars" (EROSS-2020), an interdisciplinary seminar in such as Ontology, Data Semantics, Artificial Intelligence, with the ultimate goal of contributing to the theory and practice of concepts carve reality at its joints.

Next steps (joint work with Tobias Brock and Artemis Zissis)

- Approach solves many of the conformance checking (e.g.,
- Why consider frequencies in
- Challenges currently being a
Conceptual modeling in the ’70s: top-down data design

Why does the conceptual disappear at run time?
Principles and tools for “using” the conceptual schema at run time
Ontology-based data management

Based on three main components:

- **Ontology**, a declarative, logic-based specification of the domain of interest, used as a unified, conceptual view for clients
- **Data sources**, representing external, independent, heterogeneous, storage (or, more generally, computational) structures
- **Mappings**, used to semantically link data at the sources to the ontology

Abstraction for several data management and interoperability scenarios
Data interoperability architectures

- **Data independence** (Date et al.)
- **Data integration** (Doan et al. 2012)
- **Data exchange** (Arenas et al. 2014)
- **Collaborative data sharing** (Karvounarakis et al. 2013)

- **Data independence at run time** (data access through ontology)
- **(Virtual) data integration** (data federation)
- **Data exchange/consolidation** (materialized data integration, ETL, ELT)
- **Collaborative data sharing** (P2P data integration, Peer data management)
The relationship between two nodes of the data interoperability network is specified by means of a set of mapping assertions.

**Syntax of mapping assertion**

Each mapping assertions from (source) node $S$ to (target) node $G$ has the form

$$\forall \vec{x} \ (\Phi(\vec{x}) \rightarrow \Psi(\vec{x}))$$

where

- $\Phi(\vec{x})$ is a query over $S$, whose free variables are $\vec{x}$
- $\Psi(\vec{x})$ is a query over $G$, whose free variables are from $\vec{x}$.

**Intuitive semantics of mapping assertion**

The data items in $S$ satisfying the pattern expressed as $\Phi(\vec{x})$ “match” the data items in $G$ satisfying the pattern expressed as $\Psi(\vec{x})$. 
Types of mappings

Taxonomy from data integration [L. 2002]

- **Global as view (GAV)** – $\Psi(\vec{x})$ is an **atom** $v(\vec{x})$
  
  $\forall \vec{x} (\Phi(\vec{x}) \rightarrow v(\vec{x}))$

  the element $v$ of the target node is associated with a view over the source node

- **Local as view (LAV)** – $\Phi(\vec{x})$ is an **atom** $v(\vec{x})$
  
  $\forall \vec{x} (v(\vec{x}) \rightarrow \Psi(\vec{x}))$

  the element $v$ of the source node is associated with a view over the target node

- **GLAV**
  
  $\forall \vec{x} (\Phi(\vec{x}) \rightarrow \Psi(\vec{x}))$

  a view over the source node is associated to a view over the target node
Consider the following mapping relating **Node 1** to **Node 2**:

\[
\forall x \forall y \forall z \text{ Student}(x) \land \text{Grade}(x, y, z) \rightarrow \exists w \text{ Teaches}(w, y) \land \text{Enrolled}(x, y)
\]

- **Data at Node 1**:
  
  \{ Student(Mario), Grade(Mario,IS,B), Student(Anna), Grade(Anna,AI,A) \}

After data exchange:

- **Data at Node 2** after data exchange:
  
  \{ Teaches(w_1,IS), Enrolled(Mario,IS), Teaches(w_2,Al), Enrolled(Anna,Al) \}
This talk: how to use mappings for processing queries in ontology-based data management

- **Compile time**
  - Discovery
  - Analysis (equivalence, redundancy, optimization)
  - Reasoning (inverse, composition, etc.)

- **Run time**
  - Exchanging data
  - Update propagation
  - Translation
  - **Processing queries in ontology-based data management**
1. Ontology-based data management
2. Processing queries in OBDM: answering and abstraction
3. Query answering
4. Query abstraction
5. Conclusion
Outline

1. Ontology-based data management
2. Processing queries in OBDM: answering and abstraction
3. Query answering
4. Query abstraction
5. Conclusion
An ontology-based data management (OBDM) specification [Poggi et al. 2008, L. 2018] is a triple $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$, where

- $\mathcal{O}$ is the ontology, expressed as a logical theory – here, as usual, a TBox in a Description Logic (DL)
- $\mathcal{S}$ is a data schema representing the data sources – here, a (federated) relational schema
- $\mathcal{M}$ is a set of mapping assertions, each of the form

\[ \forall \vec{x} \ (\Phi(\vec{x}) \rightarrow \Psi(\vec{x})) \]

where (in this talk we focus on conjunctive queries, but we could be more general)

- $\Phi(\vec{x})$ is a conjunctive query (SPJ query) over $\mathcal{S}$, with free variables $\vec{x}$
- $\Psi(\vec{x})$ is a conjunctive query (SPJ query) over $\mathcal{O}$, with free variables from $\vec{x}$.

An OBDM system is a pair $(\mathcal{J}, D)$, where $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$ is an OBDM specification, and $D$ is an $\mathcal{S}$-database, i.e., a source database that is legal for $\mathcal{S}$. 
Ontology-based data management specification – Example

Ontology $\mathcal{O}$ (TBox)

Employee $\sqsubseteq \exists$ worksFor
Employee $\sqsubseteq \exists$ empCode
Employee $\sqsubseteq \exists$ salary
Project $\sqsubseteq \exists$ worksFor$
eg$
Project $\sqsubseteq \exists$ projectName
$\exists$ worksFor $\sqsubseteq$ Employee
$\exists$ worksFor$
eg$ $\sqsubseteq$ Project

Federated source schema $\mathcal{S}$

$D_1[SSN: String, PrName: String]$
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

Mapping $\mathcal{M}$

$M_1$: SELECT SSN, PrName
FROM $D_1$ → Employee(pe(SSN)), Project(pr(PrName)), projectName(pr(PrName), PrName), workFor(pe(SSN), pr(PrName))

$M_2$: SELECT Code, Salary
FROM $D_2, D_3$
WHERE $D_2$.Code = $D_3$.Code → Employee(pe(SSN)), salary(pe(SSN), Salary)

$M_3$: SELECT Code, Salary
FROM $D_2, D_3$
AND SSN NOT IN (SELECT SSN FROM $D_1$) → Employee(pe(SSN)), salary(pe(SSN), Salary)
Ontology-based data management: Semantics

Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, S \rangle$ be an OBDM specification, and let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be an interpretation for the ontology $\mathcal{O}$.

**Def.: Semantics**

The semantics of $\mathcal{J}$ is given with respect to an $S$-database $D$. $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is a model of $(\mathcal{J}, D)$ if:

- $\mathcal{I}$ satisfies all axioms of $\mathcal{O}$, i.e., is a model of $\mathcal{O}$;
- $\mathcal{I}$ satisfies $\mathcal{M}$ wrt $D$, i.e., satisfies every assertion $\Phi(\bar{x}) \rightarrow \Psi(\bar{x})$ in $\mathcal{M}$ wrt $D$, which means that the sentence $\forall \bar{x} \ (\Phi(\bar{x}) \rightarrow \Psi(\bar{x}))$ is true in $\mathcal{I} \cup D$.

$(\mathcal{J}, D)$ is satisfiable or consistent, if it admits at least one model.

**Semantics of queries**

The semantics of a query $q(\bar{x})$ posed to the OBDM system $(\mathcal{J}, D)$ is given in terms of the set $\text{cert}(q, \mathcal{J}, D)$ of certain answers, i.e., those answers that hold in all the models of $(\mathcal{J}, D)$. 
Ontology-mediated query answering

A DL-knowledge base is a pair $\langle T, A \rangle$, where $T$ is a TBox (also called ontology, i.e., general axioms on concepts and relations) and $A$ is an ABox (specific axioms on individuals).

Ontology-mediated query answering, or query answering over DL-KBs

Given a DL-knowledge base $\langle T, A \rangle$, a query $q$ over such KB and a tuple $a$ of constants in $A$, check whether $a$ is a certain answer to $q$ over $\langle T, A \rangle$, i.e., if $q(a)$ is true in every model of $T \cup A$.

Let us denote by $\mathcal{M}(D)$ the ABox obtained by “applying” the mapping $\mathcal{M}$ to $D$ (i.e., by (i) computing the answer to the left-hand-side query of each mapping assertion $m$, and (ii) for each such answer adding into the ABox the “corresponding” tuple satisfying the right-hand-side query of $m$).

Proposition

If $q$ is a query posed to $\mathcal{J} = \langle O, \mathcal{M}, S \rangle$ and $D$ is an $S$-database, then $a \in \text{cert}(q, \mathcal{J}, D)$ if and only if $a$ is a certain answer to $q$ over $\langle O, \mathcal{M}(D) \rangle$. 
1. Ontology-based data management

2. Processing queries in OBDM: answering and abstraction

3. Query answering

4. Query abstraction

5. Conclusion
Two fundamental problems regarding queries

Given OBDM specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$

1. **Query answering:**
   given a query $q_\mathcal{O}$ over the ontology $\mathcal{O}$, compute a query $q_\mathcal{S}$ that captures the certain answers to $q_\mathcal{O}$ at best under $\mathcal{J}$
   
   $\xrightarrow{}$ Direct rewriting (or, Ontology-to-Source rewriting – top-down)

2. **Query abstraction:**
   given a query $q_\mathcal{S}$ over the source schema $\mathcal{S}$, compute a query $q_\mathcal{O}$ whose certain answers capture $q_\mathcal{S}$ at best under $\mathcal{J}$
   
   $\xrightarrow{}$ Reverse rewriting (or, Source-to-Ontology rewriting – bottom-up)
Usages of query answering and abstraction

1. **Query answering:**
   - Compute the certain answers of a query expressed over the ontology
   - Look for source data that are inconsistent/incomplete with respect to the ontology
   - Tell me which sources store data corresponding to instances of relevant concepts/relationships, or relevant views of such ontology elements

2. **Query abstraction:**
   - Explain the content of a data source in terms of the ontology
   - Verify if a given data service expressed over the data sources can be expressed in terms of the ontology
   - Automatically associate semantics to open data sets
Outline

1. Ontology-based data management
2. Processing queries in OBDM: answering and abstraction
3. Query answering
4. Query abstraction
5. Conclusion
Computing certain answers

Problem

Given an OBDM specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$ and a query $q_\mathcal{O}$ over the ontology $\mathcal{O}$, compute a query $q_\mathcal{S}$ over the source schema $\mathcal{S}$ that best characterizes $q_\mathcal{O}$ under $\mathcal{J}$.

The best characterization is obtained by the so-called perfect rewriting.

Formal definition of perfect rewriting

A query $q_\mathcal{S}$ over $\mathcal{S}$ is a perfect $\mathcal{O}$-to-$\mathcal{S}$ $\mathcal{J}$-rewriting of $q_\mathcal{O}$ if for every $\mathcal{S}$-database $D$, $q_\mathcal{S}^D$ coincides with the certain answers $\text{cert}(q_\mathcal{O}, \mathcal{J}, D)$.

Two basic computational problems:

- Verification (check if $q_\mathcal{S}$ is a perfect $\mathcal{O}$-to-$\mathcal{S}$ $\mathcal{J}$-rewriting of $q_\mathcal{O}$)
- Computation (compute the perfect $\mathcal{O}$-to-$\mathcal{S}$ $\mathcal{J}$-rewriting of $q_\mathcal{O}$)

Starting from [Calvanese et al. 2007], computing perfect $\mathcal{O}$-to-$\mathcal{S}$ $\mathcal{J}$-rewritings in several scenarios is one of the most studied problems in KR&R in recent years.
Direct rewriting for computing certain answer

Note that ComputerProfessor is partitioned into ComputerScientist and ComputerEngineer. Here is the ABox corresponding to $\mathcal{M}(D)$, rendered as a “knowledge graph”:

Query:

\[
\{ (x) \mid \exists y, z. \text{supervisedBy}(x, y), \text{ComputerSC}(y), \text{hates}(y, z), \text{ComputerEng}(z) \}
\]

Answer: ???
Direct rewriting for computing certain answer

Note that ComputerProfessor is partitioned into ComputerScientist and ComputerEngineer. Here is the ABox corresponding to $\mathcal{M}(D)$, rendered as a “knowledge graph”:

Query:

{ $\exists y, z. \text{supervisedBy}(x, y), \text{ComputerSC}(y), \text{hates}(y, z), \text{ComputerEng}(z) \}$

Answer: \{ john \}  To obtain this answer, we need to reasoning by cases

\rightarrow No first-order query is a perfect ontology-to-source rewriting of $q$
Complexity of conjunctive query answering in DLs

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>$AC^0$ (1)</td>
</tr>
<tr>
<td>OWL 2</td>
<td>???</td>
<td>coNP-hard (2)</td>
</tr>
</tbody>
</table>

(1) This makes it possible to scale with the data.

(2) Already for a TBox with a single disjunction (see example above).

Research question

Can we find interesting DLs for which we can always rewrite the query over the ontology into a FOL query over the source? A lot of research work/answers in the last decade!
Complexity of conjunctive query answering in DLs

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>$AC^0$ (1)</td>
</tr>
<tr>
<td>OWL 2</td>
<td>???</td>
<td>coNP-hard (2)</td>
</tr>
</tbody>
</table>

(1) This makes it possible to scale with the data.

(2) Already for a TBox with a single disjunction (see example above).

Research question

Can we find interesting DLs for which we can always rewrite the query over the ontology into a FOL query over the source? A lot of research work/answers in the last decade!
The *DL-Lite* family

*DL-Lite* is a family [Calvanese et al 2007] of DLs optimized according to the *tradeoff* between *expressive power* and *complexity* of query answering, with emphasis on *data*.

- The same complexity as relational databases.
- In fact, query answering is FOL-rewritable and hence can be delegated to a relational DB engine.

Nevertheless they have the right expressive power to capture the essential features of conceptual modeling formalisms.

*DL-Lite* provides robust foundations for *Ontology-Based Data Management*.

The *DL-Lite* family is at the basis of the *OWL 2 QL* profile of the W3C standard Web Ontology Language OWL.
Capturing basic ontology constructs in *DL-Lite*

<table>
<thead>
<tr>
<th>Modeling construct</th>
<th>DL-Lite</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA between classes</td>
<td>Student ⊑ Person</td>
</tr>
<tr>
<td>... and or relations</td>
<td>isMatherOf ⊑ isParentOf</td>
</tr>
<tr>
<td>Disjointness between classes</td>
<td>Student ⊑ ¬Professor</td>
</tr>
<tr>
<td>... and or relations</td>
<td>isMatherOf ⊑ ¬isFatherOf</td>
</tr>
<tr>
<td>Domain of relations</td>
<td>∃livesIn ⊑ Person</td>
</tr>
<tr>
<td>Range of relations</td>
<td>∃livesIn⁻ ⊑ City</td>
</tr>
<tr>
<td>Mandatory participation (<em>min card = 1</em>)</td>
<td>Person ⊑ ∃livesIn</td>
</tr>
<tr>
<td></td>
<td>City ⊑ ∃livesIn⁻</td>
</tr>
<tr>
<td>Functionality of relations (<em>max card = 1</em>)</td>
<td>(funct livesIn)</td>
</tr>
<tr>
<td>but a functional relation cannot be specialized</td>
<td>(funct livesIn⁻)</td>
</tr>
</tbody>
</table>

**Note:** *DL-Lite* distinguishes between abstract objects and data values as well (we can represent concept attributes) (ignored here).
Query answering in *DL-Lite*-based OBDM systems

In a *DL-Lite*-based OBDM specification $\langle O, M, S \rangle$

- $O$ is expressed in *DL-Lite*
- $M$ is a set of GAV mapping assertions (the right-hand side $\Psi$ is a conjunctive query (CQ) without existential variables).
- queries over $O$ are unions of conjunctive queries (UCQs).

Query answering is performed through ontology-to-source rewriting:

Given a (U)CQ $q$, OBDM specification $J = \langle O, M, S \rangle$, $S$-database $D$, we compute $\text{cert}(q, J, D)$ as follows:

1. we compute the **ontology rewriting** $r_{q,O}$ of $q$ using $O$;
2. we compute the **mapping rewriting** $r$ of $r_{q,O}$ using $M$;
3. evaluate the UCQ $r$ directly over $D$.

Correctness of this procedure shows that $r$ is the perfect ontology-to-source rewriting of $q$ wrt $J$, and shows **FOL-rewritability** of query answering in *DL-Lite*.
Query answering in \textit{DL-Lite}-based OBDM systems

\( \mathcal{S} \) contains two tables

- \( T_{\text{REG}}(\text{ID}, \text{JOB}) \)
- \( T_{\text{STUDENT}}(\text{ID}, \text{UNIVERSITY}) \)

Ontology query \( q_o \) :
\[
\{ (x) \mid \text{Person}(x) \}
\]

Ontology rewriting:
\[
\{ (x) \mid \text{Person}(x) \lor \text{Employee}(x) \lor \text{Student}(x) \}
\]

Mapping rewriting:
\[
\{ (x) \mid T_{\text{REG}}(x, y) \lor T_{\text{STUDENT}}(x, z) \}
\]
Computational complexity of query answering

Proposition

Query answering on a DL-Lite ontology-based data management system $((O, M, S), D)$ of the kind considered so far is

1. PTIME in the size of the ontology $O$ and the mappings $M$.
2. AC$^0$ in the size of the database $D$, in fact FOL-rewritable.
3. Exponential in the size of the query, more precisely NP-complete.

Precisely the complexity of evaluating CQs in plain relational DBs.

Can we go beyond DL-Lite and remain in AC$^0$?

The DLs of the DL-Lite family are essentially the maximally expressive DLs enjoying these nice computational properties.
Outline

1. Ontology-based data management
2. Processing queries in OBDM: answering and abstraction
3. Query answering
4. Query abstraction
5. Conclusion
Problem

Given an OBDM specification $J = \langle O, M, S \rangle$ and a query $q_S$ over the sources schema $S$, compute a query $q_O$ over the ontology $O$ whose certain answers best characterize $q_S$ under $J$.

Sort of reverse engineering problem, that is relevant is several tasks, e.g.,

- providing the semantics of the various data sources;
- providing the semantics of data services expressed over the sources;
- checking whether the ontology and the mapping assertions are “adequate” for answering source queries through the OBDM system;
- automatically documenting the semantics of open data sets.

The notion of abstraction (sometimes called realization or reverse rewriting) aims at addressing this problem [Cima 2017, Lutz et al. 2018, Cima et al. 2019, Cima 2020].
Perfect abstraction

Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDM specification, and let $q_S$ be a query over $\mathcal{S}$. The best characterization is obtained by the so-called perfect abstraction.

**Definition (Formal definition of abstraction)**

A query $q_\mathcal{O}$ over $\mathcal{O}$ is a perfect $\mathcal{J}$-abstraction of $q_S$, if for every $\mathcal{S}$-database $D$ such that $\langle \mathcal{J}, D \rangle$ is consistent, we have that

$$q_S^D = \text{cert}(q_\mathcal{O}, \mathcal{J}, D)$$

<table>
<thead>
<tr>
<th>Query answering</th>
<th>Query abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ontology-to-Source rewriting</strong></td>
<td><strong>Source-to-Ontology rewriting</strong></td>
</tr>
<tr>
<td>(top-down)</td>
<td>(bottom-up)</td>
</tr>
<tr>
<td><strong>input</strong>: $q_\mathcal{O}$</td>
<td><strong>input</strong>: $q_S$</td>
</tr>
<tr>
<td><strong>output</strong>: $q_S$</td>
<td><strong>output</strong>: $q_\mathcal{O}$</td>
</tr>
</tbody>
</table>

Two basic computational problems:

- Verification (check if $q_\mathcal{O}$ is a perfect $\mathcal{J}$-abstraction of $q_S$)
- Computation (compute the perfect $\mathcal{J}$-abstraction of $q_S$)
Source query $q_S$:
select ID as x from T_STUDENT

Source query $q'_S$:
select ID as x from T_REG

Perfect abstraction - Example
Perfect abstraction - Example

Source query \( q_S : \)
\[
\text{select ID as x from T\_STUDENT}
\]

Source query \( q'_S : \)
\[
\text{select ID as x from T\_REG}
\]

- perfect \( J \)-abstraction of \( q_S : \) ???
Source query $q_S$:
select ID as $x$ from T_STUDENT

Source query $q'_S$:
select ID as $x$ from T_REG

perfect $J$-abstraction of $q_S$: Student($x$)
Source query $q_S$:
select ID as x from T_STUDENT

Source query $q'_S$:
select ID as x from T_REG

- perfect $\mathcal{J}$-abstraction of $q_S$: $\text{Student}(x)$
- perfect $\mathcal{J}$-abstraction of $q'_S$: $???
Perfect abstraction - Example

Source query $q_S$:
select ID as x from T_STUDENT

Source query $q'_S$:
select ID as x from T_REG

- perfect $\mathcal{I}$-abstraction of $q_S$: Student($x$)
- perfect $\mathcal{I}$-abstraction of $q'_S$: none
The perfect abstraction for a source query may not exist.

Let \( \mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle \) be an OBDM specification, and let \( q_S \) be a query over \( \mathcal{S} \).

**Definition (Sound abstraction)**

A query \( q_\mathcal{O} \) over \( \mathcal{O} \) is a **sound** \( \mathcal{J} \)-abstraction of \( q_S \) if for every \( \mathcal{S} \)-database \( D \) such that \( \langle \mathcal{J}, D \rangle \) is consistent, we have that

\[
\text{cert}(q_\mathcal{O}, \mathcal{J}, D) \subseteq q_S^D
\]

**Definition (Complete abstraction)**

A query \( q_\mathcal{O} \) over \( \mathcal{O} \) is a **complete** \( \mathcal{J} \)-abstraction of \( q_S \) if for every \( \mathcal{S} \)-database \( D \) such that \( \langle \mathcal{J}, D \rangle \) is consistent, we have that

\[
q_S^D \subseteq \text{cert}(q_\mathcal{O}, \mathcal{J}, D)
\]
Sound and complete abstractions - Example

Source query $q_\text{S} :$
select ID as x from T_STUDENT

Source query $q'_\text{S} :$
select ID as x from T_REG

- perfect $\mathcal{J}$-abstraction of $q_\text{S}$: $\text{Student}(x)$
- perfect $\mathcal{J}$-abstraction of $q'_\text{S}$: none
Sound and complete abstractions - Example

Source query $q_S$:
select ID as x from T_STUDENT

Source query $q'_S$:
select ID as x from T_REG

- perfect $\mathcal{J}$-abstraction of $q_S$: $\text{Student}(x)$
- perfect $\mathcal{J}$-abstraction of $q'_S$: none
- complete $\mathcal{J}$-abstraction of $q'_S$: ???
- complete $\mathcal{J}$-abstraction of $q'_S$: ???
Source query $q_S$:
select ID as x from T_STUDENT

Source query $q_S'$:
select ID as x from T_REG

- perfect $J$-abstraction of $q_S$: Student($x$)
- perfect $J$-abstraction of $q_S'$: none
- complete $J$-abstraction of $q_S'$: Animal($x$)
- complete $J$-abstraction of $q_S'$: Person($x$)
Sound and complete abstractions - Example

Source query $q_S$:
select ID as $x$ from T_STUDENT

Source query $q'_S$:
select ID as $x$ from T_REG

- perfect $J$-abstraction of $q_S$: Student($x$)
- perfect $J$-abstraction of $q'_S$: none
- complete $J$-abstraction of $q'_S$: Animal($x$)
- complete $J$-abstraction of $q'_S$: Person($x$)
- sound $J$-abstraction of $q'_S$: ???
- sound $J$-abstraction of $q'_S$: ???
Sound and complete abstractions - Example

Source query $q_S$:
select ID as x from T_STUDENT

Source query $q'_S$:
select ID as x from T_REG

- perfect $\mathcal{J}$-abstraction of $q_S$: Student($x$)
- perfect $\mathcal{J}$-abstraction of $q'_S$: none
- complete $\mathcal{J}$-abstraction of $q'_S$: Animal($x$)
- complete $\mathcal{J}$-abstraction of $q'_S$: Person($x$)
- sound $\mathcal{J}$-abstraction of $q'_S$: Person($x$), University($x$)
- sound $\mathcal{J}$-abstraction of $q'_S$: Employee($x$)
Let $\mathcal{L}$ be a class of queries.

**Definition**

If $q_\varnothing \in \mathcal{L}$ is a sound $\mathcal{J}$-abstraction of $q_S$, then $q_\varnothing$ is $\mathcal{L}$-**maximally sound** if no $q' \in \mathcal{L}$ exists such that

(i) $q'$ is a sound $\mathcal{J}$-abstraction of $q_S$,

(ii) $\forall S$-database $D$ $\text{cert}(q_\varnothing, \mathcal{J}, D) \subseteq \text{cert}(q', \mathcal{J}, D)$, and

(iii) there exists an $S$-database $D$ s.t. $\text{cert}(q_\varnothing, \mathcal{J}, D) \subset \text{cert}(q', \mathcal{J}, D)$.

**Definition**

If $q_\varnothing \in \mathcal{L}$ is a complete $\mathcal{J}$-abstraction of $q_S$, then $q_\varnothing$ is $\mathcal{L}$-**minimally complete** if no $q' \in \mathcal{L}$ exists such that

(i) $q'$ is a complete $\mathcal{J}$-abstraction of $q_S$,

(ii) $\forall S$-database $D$ $\text{cert}(q', \mathcal{J}, D) \subseteq \text{cert}(q_\varnothing, \mathcal{J}, D)$, and

(iii) there exists an $S$-database s.t. $\text{cert}(q', \mathcal{J}, D) \subset \text{cert}(q_\varnothing, \mathcal{J}, D)$.
Maximal and minimal abstractions - Example

Source query $q_S$:
select ID as x from T_STUDENT

Source query $q'_S$:
select ID as x from T_REG

- perfect $\mathcal{J}$-abstraction of $q_S$: Student($x$)
- perfect $\mathcal{J}$-abstraction of $q'_S$: none
- UCQ-minimally complete $\mathcal{J}$-abstraction of $q'_S$: ???
- UCQ-maximally sound $\mathcal{J}$-abstraction of $q'_S$: ???
Maximal and minimal abstractions - Example

Source query \( q_S : \)
select ID as x from T_STUDENT

Source query \( q'_S : \)
select ID as x from T_REG

- perfect \( J \)-abstraction of \( q_S \): \text{Student}(x)
- perfect \( J \)-abstraction of \( q'_S \): none
- UCQ-minimally complete \( J \)-abstraction of \( q'_S \): \text{Person}(x)
- UCQ-maximally sound \( J \)-abstraction of \( q'_S \): \text{Employee}(x)
Can we compute the UCQ-maximally complete abstractions?

**Theorem**

The verification problem for complete abstractions is NP-complete.

What about the computation problem? We now focus on the problem of computing the **UCQ-maximally complete abstraction** of a $q_S$ over the source schema $S$ with respect to an OBDM specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, S \rangle$, where

- $q_S$ is a CQ
- $\mathcal{O}$ a DL-Lite$^R$ (therefore, no functionality assertions) ontology, and
- $\mathcal{M}$ a set of GLAV mapping assertions of the form

$$\forall \vec{x} \ \forall \vec{y} \ (\Phi_S(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \ \Psi_O(\vec{x}, \vec{z}))$$

where $\Phi_S(\vec{x})$ is a CQ over $S$ and $\Psi_O$ is a CQ over the ontology $\mathcal{O}$.

Note that computing such rewriting solves also the problem of computing the **perfect abstraction** (it is sufficient to check the result for soundness).
How to compute it: basic idea

There is a strict correlation between CQs and relational databases. Given a CQ $q$ over a schema $S$ it is possible to construct in linear time an $S$-database $D_q$ that fully captures $q$, and vice versa:

- every constant in $q$ becomes a value in $D_q$;
- every variable in $q$ becomes a labeled null in $D_q$;
- every atom $R_i(\vec{u}) \in q$ becomes a fact (tuple) in $D_q$.

Example

Let $S$ contain the tables $\text{Tab1}(id,\text{city})$ and $\text{Tab2}(id,\text{city})$, and consider the following CQ $q_S$ over $S$

$$q_S(x) \leftarrow \text{Tab1}(x, y), \text{Tab2}(\text{‘sara’}, y)$$

The $S$-database $D_{q_S}$ associated to the query $q_S$ is:

$$\text{Tab1}(x,y), \text{Tab2}(\text{‘sara’}, y)$$

where $x, y$ are labeled nulls.
Roughly speaking, the $S$-database $D_{qs}$ associated to $qs$ is representative of those instances of $S$ on which the evaluation of $qs$ is non-empty.

Given the OBDM specification $J = \langle O, M, S \rangle$, we denote by $M(D_{qs})$ the set of $O$-facts (ABox, in DL terminology) obtained by chasing $D_{qs}$ with the mapping assertions in $M$.

**Intuition**

The query $q_O$ over the ontology corresponding to the UCQ-minimally complete $J$-abstraction of $qs$ is the query corresponding to $M(D_{qs})$. 

\[
\begin{align*}
q_S & \rightarrow D_{qs} \rightarrow M(D_{qs}) \rightarrow q_O \\
\text{compute the} & \quad \text{compute the} & \quad \text{compute the} & \quad \text{compute the} & \quad \text{compute the} \\
\text{associated DB} & \quad \text{chase} & \quad \text{corresponding query} & 
\end{align*}
\]
Example

Suppose to have the following mapping:

\[
m_1: \quad \text{Tab1}(x,y) \leadsto \text{Person}(x), \text{livesIn}(x,y) \\
m_2: \quad \text{Tab2}(x,y) \leadsto \text{Person}(x), \text{worksIn}(x,y)
\]

and consider the instance \( D_{qs} \) of \( S \) associated to the query \( q_S \):

\[
\text{Tab1}(x,y), \quad \text{Tab2}(\text{‘sara’},y),
\]

It is easy to see that \( \mathcal{M}(D_{qs}) \) is the ABox containing the following assertions:

\[
\text{Person}(x), \text{livesIn}(x,y), \text{Person}(\text{‘sara’}), \text{worksIn}(\text{‘sara’},y)
\]

From \( \mathcal{M}(D_{qs}) \) we can construct the following query over the ontology, denoting all the Persons living in the same city where the person ‘sara’ works

\[
q_{\mathcal{O}}(x) \leftarrow \text{Person}(x), \text{livesIn}(x,y), \text{Person}(\text{‘sara’}), \text{worksIn}(\text{‘sara’},y)
\]
We denote by $\top$ the special conjunctive query over $\mathcal{O}$ that is true for every tuple.

**Algorithm** FindMinimallyCompleteAbstraction

**Input:** a $DL$-Lite R OBDM specification $\langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$

UCQ $q_{S} = q_{S}^{1}(\vec{t}_{1}) \cup \ldots \cup q_{S}^{n}(\vec{t}_{n})$ over $S$

**Output:** UCQ $q_{\mathcal{O}}$ over $\mathcal{O}$

**begin**

1. $q_{\mathcal{O}} := \{ \vec{t}_{1} \mid \mathcal{M}(q_{S}^{1}(\vec{t}_{1})), \top(\vec{t}_{1}) \} \cup \ldots \cup \{ \vec{t}_{n} \mid \mathcal{M}(q_{S}^{n}(\vec{t}_{n})), \top(\vec{t}_{n}) \}$

2. return $q_{\mathcal{O}}$

**end**

Informally, the algorithm computes the output query as union of CQs obtained by simply applying the mapping $\mathcal{M}$ to each CQ $q_{S}^{i}$ in $q_{S}$, using $\top$ to bind the distinguished variables of the output query that do not appear in $\mathcal{M}(q_{S}^{i})$. 
Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$ be a DL-Lite\textsubscript{R} OBDM specification, and let $q_S$ be a CQ over $\mathcal{S}$.

- **FindMinimallyCompleteAbstraction**($\mathcal{J}, q_S$) terminates and runs in:
  
  (i) $\text{PTime}$ in the size of $q_S$;
  
  (ii) $\text{PTime}$ in the size of $\mathcal{O}$;
  
  (iii) $\text{ExpTime}$ in the size of $\mathcal{M}$.

- The query returned by **FindMinimallyCompleteAbstraction**($\mathcal{J}, q_S$) is a UCQ-minimally complete $\mathcal{J}$-abstraction of $q_S$.

- The UCQ-minimally complete $\mathcal{J}$-abstraction of $q_S$ is unique up to logical equivalence, and it can be expressed as a CQ if $q_S$ is a CQ.

- A perfect $\mathcal{J}$-abstraction of $q_S$ exists in the class of positive queries if and only if the query $q_\mathcal{O}$ returned by **FindMinimallyCompleteAbstraction**($\mathcal{J}, q_S$) is a sound $\mathcal{J}$-abstraction of $q_S$, in which case $q_\mathcal{O}$ is the perfect $\mathcal{J}$-abstraction of $q_S$. 
Computing UCQ-maximally sound $\mathcal{J}$-abstractions is more involved.

**Theorem**

The verification problem for sound abstractions is $\Pi_{2}^{p}$-complete.

**Theorem**

UCQ-maximally sound $\mathcal{J}$-abstractions of a query $q_S$ may not exist if the OBDM specification $\mathcal{J}$ has one of the following features:

1. **disjointness axioms** in the ontology;
2. inclusion axioms with $\exists R$ as right-hand side in the ontology;
3. **LAV** mapping assertions, even without joins involving existential variables in the right-hand side;
4. **non-pure GAV** mapping assertions;
5. $q_S$ has joins on existential variables.

We have devised an algorithm for computing UCQ-maximally sound $\mathcal{J}$-abstractions in the restricted setting with none of the above features.
A GAV mapping is pure if for each mapping assertion, the variables in the right-hand-side query are different. Consider the following OBDA specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ with non-pure GAV mapping:

- $\mathcal{O} = \emptyset$,
- $\mathcal{S} = \{ s_1, s_2 \}$,
- $\mathcal{M} = \{ s_1(x_1, x_2) \rightarrow R(x_1, x_2), \quad s_2(x) \rightarrow R(x, x) \}$,

and the following $q_S = \{ (x_1, x_2) \mid s_1(x_1, x_2) \}$.

$q'_\mathcal{O} = \{ (x_1, x_2) \mid R(x_1, x_2) \}$ is the UCQ-minimally complete $\mathcal{J}$-rewriting of $q_S$, but is not sound. Intuitively, the infinite query:

$$q_\varnothing = \bigcup_{a, b \in \text{const} \text{ with } a \neq b} \{ (a, b) \mid R(a, b) \}$$

is the maximally sound $\mathcal{J}$-abstraction of $q_S$ in the class of positive queries, but is not a UCQ → no UCQ-maximally sound $\mathcal{J}$-abstraction of $q_\varnothing$ exists.
1 Ontology-based data management
2 Processing queries in OBDM: answering and abstraction
3 Query answering
4 Query abstraction
5 Conclusion
Future work on query abstraction

A notable challenge: can we obtain better sound abstractions if we express abstractions with languages going beyond UCQ?

Source query $q'_S$:
select ID as x from T_REG

- UCQ-maximally sound $\mathcal{J}$-abstraction of $q'_S$: Employee($x$)
- A better sound $\mathcal{J}$-abstraction of $q'_S$: Person($x$), $\neg$ KStudent($x$), corresponding to the non-monotonic query in EQL-lite asking for all persons that are not known to be students.
Future work on query abstraction

More challenges:

- The existence problem.
- Investigate ontology query languages beyond UCQ (such as EQL-lite).
- Single out more restricted settings guaranteeing existence of UCQ-maximally sound abstractions.
- Going beyond $DL-Lite_R$.
- We have assumed that the query language used to express $q_S$ is the language of CQs. This is too limited: real applications often requires aggregation, negation, universal quantification, etc. Can we compute abstractions of queries with such operators?
- Study specific applications of abstraction, and try to specialize the corresponding techniques.
- Investigate possible use of query abstraction in explanation of classifiers.

Research themes in HOPE, a national MIUR-PRIN project with Sapienza University of Rome, Free University of Bozen/Bolzano, Polytechnic University of Milan, University of Milan, and University of Cagliari.
Joint work with(*)

- Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Antonella Poggi, Riccardo Rosati, *and many others (query answering)*
- Gianluca Cima, Antonella Poggi *(query abstraction)*

Thank you for your attention!

(*) *My heart is not weary, it’s light and it’s free; I’ve got nothin’ but affection for all those who’ve sailed with me – Bob Dylan*