Managing open data through ontologies

Maurizio Lenzerini

joint work with Gianluca Cima, Antonella Poggi

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

ISIP 2019

13th Int. Workshop on Information Search, Integration, and Personalization
Heraklion, Greece, May 9-10, 2019
What is Open Data?

“Open data is data that anyone can access, use or share.”

Why Open Data?

- interoperability
- transparency in government
- improving public services
- creating social and commercial value
- essential element for the data-driven society

---

1Open Data Institute (https://theodi.org/)
Current practices for publishing Open Data focus essentially on providing the dataset (extensional information), often in simple format as CSV. Dataset are mostly documented through description in natural language. As a consequence, the semantics of datasets is not formally expressed in a machine-readable form.

In this talk we explore the use of the paradigm of **Ontology-based Data Management (OBDM)** as a means for publishing open data that are

- high-quality
- semantically annotated
Outline

1. Ontology-based data management: the framework
2. Ontology-based open data
3. Ontology-to-Source rewriting
4. Source-to-Ontology rewriting
5. Conclusion
Outline

1. Ontology-based data management: the framework
2. Ontology-based open data
3. Ontology-to-Source rewriting
4. Source-to-Ontology rewriting
5. Conclusion
Ontology-based data management: architecture

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
An ontology-based data management (OBDM) specification [Poggi et al 2008] is a triple $\mathcal{J} = \langle O, M, S \rangle$, where

- $O$ is the ontology, expressed as a logical theory (here, is a TBox in a Description Logic)
- $S$ is a (federated) relational database schema representing the data sources
- $M$ is a set of mapping assertions, each of the form
  
  $$\Phi(\vec{x}) \sim \Psi(\vec{x})$$

where

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

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

Ontology $\mathcal{O}$ (TBox)

- Employee ⫋ ∃worksFor
- Employee ⫋ ∃empCode
- Employee ⫋ ∃salary
- Project ⫋ ∃worksFor
- Project ⫋ ∃projectName
- ∃worksFor ⫋ Employee
- ∃worksFor ⫋ Project

Federated schema of the DB $S$

- $D_1[SSN: \text{String}, PrName: \text{String}]$
  - Employees and Projects they work for
- $D_2[Code: \text{String}, Salary: \text{Int}]$
  - Employee’s Code with salary
- $D_3[Code: \text{String}, SSN: \text{String}]$
  - Employee’s Code with SSN

Mapping $\mathcal{M}$

$M_1$: SELECT SSN,PrName
FROM $D_1$

$M_2$: SELECT SSN,Salary
FROM $D_2$, $D_3$

$M_1$ $\leadsto$
Employee(pers(SSN)),
Project(proj(PrName)),
projectName(proj(PrName), PrName),
workFor(pers(SSN), proj(PrName))

$M_2$ $\leadsto$
Employee(pers(SSN)),
salary(pers(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 a legal instance $D$ of $S$. $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ 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 $\Phi(\vec{x}) \leadsto \Psi(\vec{x})$ in $\mathcal{M}$ wrt $D$, which means that the sentence $\forall \vec{x} \ (\Phi(\vec{x}) \rightarrow \Psi(\vec{x}))$ is true in $\mathcal{I} \cup D$.

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

**Certain answer**

Query processing over the OBDM system $(\mathcal{J}, D)$ amounts to finding the **certain answers** $\text{cert}(q, \mathcal{J}, D)$ to a query $q(\vec{x})$, i.e., those answers that hold in **all the models** of $(\mathcal{J}, D)$.
Once we have defined an **OBDM** specification over our data sources, we can exploit it for the tasks needed for publishing **open data**, at least in two scenarios:

1. **Top-down**: we want to publish a dataset and we know how to specify its content in terms of the ontology (open data publication and annotation)
2. **Bottom-up**: we have an existing dataset, and we want to produce a semantic description of it (open data annotation)
Scenario 1: The top-down approach

⇒ If the IT-expert is familiar with OBDM, the dataset to publish is specified by referring to the domain’s elements (the ontology).

Example: Mr White (the manager) wants to publish data (number of vehicles per city) regarding vehicles registered this year in all those regions where the number of citizens is greater than 4 mln. Mr. White himself, or Johnny (the OBDM-expert), know how to formulate the request in terms of the ontology.
The way to proceed is:

1. express the request of the dataset we want to open in terms of a query over the ontology;
2. evaluate such query over the OBDM system (compute the certain answers);
3. express the result in RDF by using the query expression and the ontology;
4. publish the ontology, the query, and the dataset.
Scenario 2: The bottom-up approach

The dataset is already published, and the query over the source used to produce it is known; we now want to automatically associate to it a semantic annotation.

The way to proceed is:

1. given the query over the data source used to produce the dataset, a mechanism is needed in order translate the source query into the ontology, so as to derive a query over the ontology;
2. use such query as a semantic annotation of the dataset.
What do we need in the two scenarios

Once we have the OBDM specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$, we need

1. **In scenario 1:**
   a method that, given a query $q_\mathcal{O}$ over the ontology $\mathcal{O}$, computes a query $q_\mathcal{S}$ such that, for every database $D$ for $\mathcal{S}$, $q^D_\mathcal{S}$ coincides with the certain answers $\text{cert}(q_\mathcal{O}, \mathcal{J}, D)$

   $\rightarrow$ **Ontology-to-Source query rewriting**

2. **In scenario 2:**
   a method that, given a query $q_\mathcal{S}$ over the source schema $\mathcal{S}$, computes a query $q_\mathcal{O}$ such that, for every database $D$ for $\mathcal{S}$, $\text{cert}(q_\mathcal{O}, \mathcal{J}, D)$ coincides with $q^D_\mathcal{S}$

   $\rightarrow$ **Source-to-Ontology query rewriting**
Outline

1. Ontology-based data management: the framework
2. Ontology-based open data
3. Ontology-to-Source rewriting
4. Source-to-Ontology rewriting
5. Conclusion
Ontology-to-Source rewriting

Depending of the expressive power of the query, ontology and mapping language, the problem may become very involved.

Note that ComputerProfessor is partitioned into ComputerScientist and ComputerEngineer.

Query:

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

Answer: ???
Ontology-to-Source rewriting

Depending of the expressive power of the query, ontology and mapping language, the problem may become very involved.

Note that ComputerProfessor is partitioned into ComputerScientist and ComputerEngineer.

Query:
\[ q(x) \leftarrow \exists y, z. \text{supervisedBy}(x, y), \text{ComputerSC}(y), \text{hates}(y, z), \text{ComputerEng}(z) \]

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

\[ \rightarrow \text{No First-order query is an Ontology-to-Source rewriting of } q \]
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).

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).

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><em>Student</em> ⊑ <em>Person</em></td>
</tr>
<tr>
<td>... and or relations</td>
<td><em>isMatherOf</em> ⊑ <em>isParentOf</em></td>
</tr>
<tr>
<td>Disjointness between classes</td>
<td><em>Student</em> ⊑ ¬<em>Professor</em></td>
</tr>
<tr>
<td>... and or relations</td>
<td><em>isMatherOf</em> ⊑ ¬<em>isFatherOf</em></td>
</tr>
<tr>
<td>Domain of properties</td>
<td>∃<em>livesIn</em> ⊑ <em>Person</em></td>
</tr>
<tr>
<td>Range of properties</td>
<td>∃<em>livesIn</em> ⊑ <em>City</em></td>
</tr>
<tr>
<td>Mandatory participation (<em>min card = 1</em>)</td>
<td><em>Person</em> ⊑ ∃<em>livesIn</em></td>
</tr>
<tr>
<td></td>
<td><em>City</em> ⊑ ∃<em>livesIn</em>−</td>
</tr>
<tr>
<td>Functionality of relations (<em>max card = 1</em>)</td>
<td>(funct <em>livesIn</em>)</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 $\mathcal{J} = \langle O, M, S \rangle$, database $D$, we compute $\text{cert}(q, \mathcal{J}, D)$ 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 **FOL-rewritability** of query answering in *DL-Lite*
Proposition

Query answering on a \textit{DL-Lite} ontology-based data management system $(\langle \mathcal{O}, \mathcal{M}, S \rangle, D)$ of the kind considered so far is

1. \textbf{\textsc{PTime}} in the size of the ontology $\mathcal{O}$ and the mappings $\mathcal{M}$.
2. \textbf{\textsc{AC}^0} in the size of the database $D$, in fact FOL-rewritable.
3. Exponential in the size of the query, more precisely \textbf{NP-complete}.

This is not bad, since this is precisely the complexity of evaluating CQs in plain relational DBs.

Can we go beyond \textit{DL-Lite} and remain in \textbf{\textsc{AC}^0}?

The DLs of the \textit{DL-Lite} family are essentially the maximally expressive DLs enjoying these nice computational properties.
1. Ontology-based data management: the framework
2. Ontology-based open data
3. Ontology-to-Source rewriting
4. Source-to-Ontology rewriting
5. Conclusion
Problem

Given an OBDM specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, S \rangle$ and a query $q_S$ over the sources schema $S$, which query $q_O$ over the ontology $\mathcal{O}$ best characterizes $q_S$ under $\mathcal{J}$?

Note that this problem is relevant also for other tasks related to the management of the information system of an organization, e.g.,

- the task of providing the semantics of the various data sources;
- the task of providing the semantics of data services expressed over the sources (reverse engineering);
- checking whether the ontology and the mapping assertions are “adequate” for answering source queries through the OBDM system.
Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, S \rangle$ be an OBDM specification, and let $q_S$ be a query over $S$.

**Definition (Perfect S-to-O rewriting)**

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

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

<table>
<thead>
<tr>
<th>Ontology-to-Source rewriting</th>
<th>Source-to-Ontology rewriting</th>
</tr>
</thead>
<tbody>
<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>
Example

Consider the OBDM specification \( \mathcal{J} = (\mathcal{O}, \mathcal{M}, \mathcal{S}) \), where

- \( \mathcal{O} = \{ \text{Student} \sqsubseteq \text{Person}, \text{Employee} \sqsubseteq \text{Person}, \text{Person} \sqsubseteq \text{Animal} \} \)
  - Animal \sqsubseteq \neg \text{University} \}

- \( \mathcal{S} \) contains two tables
  - \( \text{T_REG(ID,JOB)} \)
  - \( \text{T_STUDENT(ID,UNIVERSITY)} \)

- \( \mathcal{M} \) is as follows:
  - \( m_1: \text{select ID as } x \text{ from } \text{T_REG} \rightsquigarrow \text{Person}(x) \)
  - \( m_2: \text{select ID as } x \text{ from } \text{T_REG} \text{ where JOB = 'e'} \rightsquigarrow \text{Employee}(x) \)
  - \( m_3: \text{select ID as } x \text{ from } \text{T_STUDENT} \rightsquigarrow \text{Student}(x) \)
Source query $Q_S$:
select ID as x from T_REG

Source-to-Ontology rewriting - Example

T_REG T_STUDENT

JOB = 'e'

Animal

Person

Employee

Student

Maurizio Lenzerini
Managing open data through ontologies
Source query $Q_S$:
select ID as x from T_REG

- perfect S-to-O $\mathcal{J}$-rewriting of $Q_S$: none
Source-to-Ontology rewriting - Example

**Source query** $Q_S$:
select ID as x from T_REG

**Source query** $Q'_S$:
select ID as x from T_STUDENT

JOB = ‘e’
Source query $Q_S$: 
select ID from $T_{\text{REG}}$

Source query $Q'_S$: 
select ID as x from $T_{\text{STUDENT}}$

- **perfect S-to-O $\mathcal{J}$-rewriting of $Q_S$: none**
- **perfect S-to-O $\mathcal{J}$-rewriting of $Q'_S$: Student(x)**
Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, S \rangle$ be an OBDM specification, and let $q_S$ be a query over $S$.

**Definition (Sound S-to-O rewriting)**

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

$$\text{cert}(q_\mathcal{O}, \mathcal{J}, D) \subseteq q^D_S$$

**Definition (Complete S-to-O rewriting)**

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

$$q^D_S \subseteq \text{cert}(q_\mathcal{O}, \mathcal{J}, D)$$
Source query $Q_S$:
select ID as x from T_REG

Source query $Q'_S$:
select ID as x from T_STUDENT

JOB = 'e'
Source-to-Ontology rewritings - Example

Source query $Q_S$:
select ID as x from T_REG

Source query $Q'_S$:
select ID as x from T_STUDENT

- complete S-to-O $J$-rewriting of $Q_S$: Animal($x$)
- complete S-to-O $J$-rewriting of $Q_S$: Person($x$)
- sound S-to-O $J$-rewriting of $Q_S$: Employee($x$)
- sound S-to-O $J$-rewriting of $Q_S$: Employee($x$), University($x$)
- perfect S-to-O $J$-rewriting of $Q_S$: none
- perfect S-to-O $J$-rewriting of $Q'_S$: Student($x$)
Minimal and maximal S-to-O rewritings

Let \( \mathcal{L} \) be a class of queries.

**Definition**

If \( q_\mathcal{O} \in \mathcal{L} \) is a sound S-to-\( \mathcal{O} \) \( \mathcal{I} \)-rewriting of \( q_S \), then \( q_\mathcal{O} \) is \( \mathcal{L} \)-maximally sound if no \( q'_\mathcal{O} \in \mathcal{L} \) exists such that (i) \( q'_\mathcal{O} \) is a sound S-to-\( \mathcal{O} \) \( \mathcal{I} \)-rewriting of \( q_S \), (ii) \( \text{cert}_{q_\mathcal{O},\mathcal{I}} \subseteq \text{cert}_{q'_\mathcal{O},\mathcal{I}} \), and (iii) there exists an \( S \)-database s.t. \( \text{cert}_{D,q_\mathcal{O},\mathcal{I}} \subset \text{cert}_{D,q'_\mathcal{O},\mathcal{I}} \).

**Definition**

If \( q_\mathcal{O} \in \mathcal{L} \) is a complete S-to-\( \mathcal{O} \) \( \mathcal{I} \)-rewriting of \( q_S \), then \( q_\mathcal{O} \) is \( \mathcal{L} \)-minimally complete if no \( q'_\mathcal{O} \in \mathcal{L} \) exists such that (i) \( q'_\mathcal{O} \) is a complete S-to-\( \mathcal{O} \) \( \mathcal{I} \)-rewriting of \( q_S \), (ii) \( \text{cert}_{q'_\mathcal{O},\mathcal{I}} \subseteq \text{cert}_{q_\mathcal{O},\mathcal{I}} \), and (iii) there exists an \( S \)-database s.t. \( \text{cert}_{D,q'_\mathcal{O},\mathcal{I}} \subset \text{cert}_{D,q_\mathcal{O},\mathcal{I}} \).
Source query $Q_S$:
select ID as x from T_REG

Source query $Q'_S$:
select ID as x from T_STUDENT

- UCQ-minimally complete S-to-O $\mathcal{J}$-rewriting of $Q_S$: $\text{Person}(x)$
- UCQ-maximally sound S-to-O $\mathcal{J}$-rewriting of $Q_S$: $\text{Employee}(x)$
- perfect S-to-O $\mathcal{J}$-rewriting of $Q_S$: none
- perfect S-to-O $\mathcal{J}$-rewriting of $Q'_S$: $\text{Student}(x)$
Can we compute the UCQ-maximally complete $S$-to-$O$ rewriting?

We now focus on the problem of computing the **UCQ-maximally complete $S$-to-$O$ rewriting** of a $q_S$ over the source schema $S$ with respect to an OBDM specification $J = \langle O, M, S \rangle$, where

- $q_S$ is a **CQ**
- $O$ a **DL-Lite** ontology, and
- $M$ 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 $O$.

Note that computing such rewriting solves also the problem of computing the perfect rewriting (it is sufficient to check the result for soundness)
There is a very strict correlation between CQs and database. Given a CQ $q$ over a schema $S$ it is possible to construct in linear time a database $D_q$ of $S$ 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 $S$

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

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

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

where $x, y$ are labeled nulls.
How to compute it: basic idea (cont’d)

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

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

Intuition

The query $q_{\mathcal{O}}$ over the ontology corresponding to the UCQ-minimally complete $\mathcal{J}$-rewriting of $qs$ is the query associated to $\mathcal{M}(D_{qs})$.

\[
\begin{align*}
Q_{S} & \quad \rightarrow \quad D_{Q_{S}} \quad \rightarrow \quad \mathcal{M}(D_{Q_{S}}) \quad \rightarrow \quad Q_{\mathcal{O}} \\
\text{compute the} & \quad \text{compute the} & \quad \text{compute the} & \quad \text{compute the} \\
\text{associated DB} & \quad \text{chase} & \quad \text{associated query} & \\
\end{align*}
\]
Example

Suppose to have the following mapping:

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

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

\[ \text{Tab1}(x, y), \text{Tab2('sara', y)}, \]

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

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

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

\[ Q_{\mathcal{O}}(x) \Leftarrow \text{Person}(x), \text{livesIn}(x, y), \text{Person('sara')}, \text{worksIn('sara', y)} \]
Algorithm: **FindMinimallyCompleteRewriting**

**Input:** a DL-Lite OBDM specification \( \langle O, M, S \rangle \), a CQ \( q_S \) over \( S \)

**Output:** a CQ \( q_O \) over \( O \)

1. Compute \( D_{q_s} \) from \( q_s \) (i.e., the database, possibly with incomplete information, associated to the query \( q_s \)).

2. Chase \( D_{q_s} \) wrt \( M \) to produce an ABox \( A \) with variables.

3. If \( A = \emptyset \) (i.e., no atoms in \( A \)), then return the query \( \{ \text{tup}(q_s) \mid \top(\text{tup}(q_s)) \} \).

4. Chase \( A \) wrt \( \neg Q_1^{\neq} \); if the chase fails, then return the query \( \{ \text{tup}(Q_s) \mid \bot(\text{tup}(Q_s)) \} \); otherwise, let \( A' \) be the instance produced, and let \( \psi \) be the set of equality applied to the variables in \( D_{q_s} \) by the chase.

5. Evaluate \( q_0^{\neq} \) over \( A' \); if the answer is \( \{ () \} \) (i.e., \( J \models q_0^{\neq} \)), then return the query \( \{ \text{tup}(q_s) \mid \bot(\text{tup}(q_s)) \} \); otherwise, let \( q_{A'} \) be the boolean conjunctive query associated to \( A' \).

6. Return \( \{ \psi(\text{tup}(q_s)) \mid Q_{A'}(\psi(\text{tup}(Q_s))) \land \top(\overline{w}) \} \), where \( \overline{w} \) is the tuple composed by all terms in \( \psi(\text{tup}(q_s)) \) not appearing in \( A' \).
Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, S \rangle$ be a DL-Lite OBDM specification, and let $q_S$ be a CQ over $S$.

- **FindMinimallyCompleteRewriting$(\mathcal{J}, q_S)$** 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 **FindMinimallyCompleteRewriting$(\mathcal{J}, q_S)$** is a **UCQ-minimally complete S-to-O $\mathcal{J}$-rewriting** of $q_S$.

- The UCQ-minimally complete S-to-O $\mathcal{J}$-rewriting of $q_S$ is **unique** up to logical equivalence, and can be expressed as a conjunctive query.

- A perfect S-to-O $\mathcal{J}$-rewriting of $q_S$ exists if and only if the query $q_O$ returned by **FindMinimallyCompleteRewriting$(\mathcal{J}, q_S)$** is a sound S-to-O $\mathcal{J}$-rewriting of $q_S$, in which case $q_O$ is the perfect S-to-O $\mathcal{J}$-rewriting of $q_S$. 

1. Ontology-based data management: the framework
2. Ontology-based open data
3. Ontology-to-Source rewriting
4. Source-to-Ontology rewriting
5. Conclusion
Future work

We argue that OBDM provides a suitable formal basis for a new, well-founded methodology for open data publishing, and that in this context, S-to-O rewriting is an interesting notion.

Many challenges remain, for example:

- Studying **UCQ-maximally sound** S-to-O rewritings.
- If we relax the assumption that $\mathcal{L} = \text{UCQ}$, can we have/compute more accurate S-to-O rewritings?
- We have assumed that the query language used to express $q_S$ is the language of CQs. This is too limited: Open data often requires aggregation, negation, universal quantification, etc. Can we compute the S-to-O rewriting of queries with such operators?
Source query $q_S$:
select ID as x from T_REG

- UCQ-maximally sound S-to-O $\mathcal{J}$-rewriting of $q_S$: empty query
- “best” sound s-to-t rewriting of $q_S$?
Thank you for your attention!