Query Answering in DL-Lite with Datatypes: A Non-Uniform Approach

February 2, 2017

André Hernich, Julio Lemos, Frank Wolter
In Ontology-Based Data Management,

- data is stored in a database $A$ (an ‘ABox’, in the DL setting),
- intensional knowledge is stored in an ontology $\mathcal{T}$ (a ‘TBox’),
- queries against $A$ are answered taking $\mathcal{T}$ into account.

One of the main aims in OBDM: **tractable** query answering.
Suppose we have a company ontology/database:

\[ T = \{ \text{Employee} \equiv \exists \text{hasEmployer}, \text{Manager} \sqsubseteq \text{Employee} \} \]

\[ A = \{ \text{Employee}(Anna), \text{Manager}(Barbara), \text{hasEmployer}(Cid, IBM) \} \]

Since ontologies use open world semantics, we look for answers that hold in all models. Suppose we query \( A \) w.r.t. \( T \):

\[ q(x) \leftarrow \text{Employee}(x) \]

We will get as answers Anna, Barbara and Cid.
Given a DL-Lite TBox $\mathcal{T}$ and a conjunctive query $q$:

Query evaluation of $q$ w.r.t. $\mathcal{T}$

Input: an ABox $\mathcal{A}$, a tuple $\bar{c}$

Question: $\mathcal{T}, \mathcal{A} \models q(\bar{c})$?

Complexity of query answering with $\mathcal{T}, q$ fixed is called data complexity (here the default). Otherwise combined complexity.

This problem is in PTIME (actually FO-rewritable).
We add datatypes $\mathcal{D} = (D, R_1, \ldots, R_n)$ to ontologies and queries. E.g. fix $\mathcal{D} = (\mathbb{Z}, ((a, b) \mid a, b \in \mathbb{Z} \cup \{\infty, -\infty\}))$, set:

$$\mathcal{T'} = \mathcal{T} \cup \{\text{Employee} \sqsubseteq \exists \text{salary}, \text{Manager} \sqsubseteq \exists \text{salary} . (40000, \infty)\}$$

Query:

$$q(x) \leftarrow \text{Employee}(x), \text{salary}(x, v), v \in (40000, \infty) ,$$

and then Barbara is the answer!

Query answering with such queries is NP-hard for DL-Lite: [Savkovic-Calvanese 2012], [Artale-Ryzhikov-Kontchakov 2012].
We go further and allow datatypes with binary predicates. E.g. for $\mathcal{T}'$ and $\mathcal{A}$ as above, add the predicate “$<$” to $\mathcal{D}$ and consider the query

$q(x, y) \leftarrow \text{Employee}(x), \text{Employee}(y), \text{salary}(x, v), \text{salary}(y, w), v < w$
We go further and allow datatypes with binary predicates. E.g. for $\mathcal{T}'$ and $\mathcal{A}$ as above, add the predicate “$<$” to $\mathcal{D}$ and consider the query

$q(x, y) \leftarrow \text{Employee}(x), \text{Employee}(y), \text{salary}(x, v), \text{salary}(y, w), v < w$

It turns out...
OBDM WITH DATATYPES

We go further and allow datatypes with binary predicates. E.g. for $\mathcal{T}'$ and $\mathcal{A}$ as above, add the predicate “$<$” to $\mathcal{D}$ and consider the query

$q(x, y) \leftarrow \text{Employee}(x), \text{Employee}(y), \text{salary}(x, v), \text{salary}(y, w), v < w$

It turns out...

Theorem

Query answering in DL-Lite is undecidable in combined complexity for the datatypes $(\mathbb{Z}, \neq), (\mathbb{Z}, <), (\mathbb{Z}, \leq), (\mathbb{Q}, \neq), (\mathbb{Q}, <)$. 
OBDM WITH DATATYPES

We go further and allow datatypes with binary predicates. E.g. for $\mathcal{T}'$ and $\mathcal{A}$ as above, add the predicate “$<$” to $\mathcal{D}$ and consider the query

$$q(x, y) \leftarrow \text{Employee}(x), \text{Employee}(y), \text{salary}(x, v), \text{salary}(y, w), v < w$$

It turns out...

Theorem

Query answering in DL-Lite is undecidable in combined complexity for the datatypes $(\mathbb{Z}, \neq), (\mathbb{Z}, <), (\mathbb{Z}, \leq), (\mathbb{Q}, \neq), (\mathbb{Q}, <)$.

In face of these negative results, our goal is a classification:

![Classification Diagram]

Tractable/intractable based on how datatype atoms occur in $\mathcal{T}$ and $q$. 
DL-Lite and our extension

**DL-Lite**: Fix a datatype $\mathcal{D}$. In DL-Lite we allow concepts $B := A \mid \exists r \mid \exists U$ where $r$ can be a role or an inverse role and $U$ an attribute over $\mathcal{D} = (D, R_1, \ldots, R_n)$.

The language supports axioms of the form $B_1 \sqsubseteq (\neg)B_2$. 
DL-Lite: Fix a datatype $\mathcal{D}$. In DL-Lite we allow concepts $B := A \mid \exists r \mid \exists U$ where $r$ can be a role or an inverse role and $U$ an attribute over $\mathcal{D} = (D, R_1, \ldots, R_n)$.

The language supports axioms of the form $B_1 \sqsubseteq (\neg)B_2$.

We then extend DL-Lite with (non-negated) concepts on the right-hand side restricting the range of attributes $U$:

- $B \sqsubseteq \exists U. \varphi(x)$
- $B \sqsubseteq \forall U. \varphi(x)$

with $\varphi(x) \leftarrow R_1(\bar{z}_1), \ldots, R_m(\bar{z}_m)$. 
**DL-Lite and our extension**

**DL-Lite**: Fix a datatype $\mathcal{D}$. In DL-Lite we allow concepts $B := A \mid \exists r \mid \exists U$ where $r$ can be a role or an inverse role and $U$ an attribute over $\mathcal{D} = (D, R_1, \ldots, R_n)$.

The language supports axioms of the form $B_1 \sqsubseteq (\neg)B_2$.

We then extend DL-Lite with (non-negated) concepts on the right-hand side restricting the range of attributes $U$:

- $B \sqsubseteq \exists U.\varphi(x)$
- $B \sqsubseteq \forall U.\varphi(x)$

with $\varphi(x) \leftarrow R_1(\overline{z}_1), \ldots, R_m(\overline{z}_m)$.

**Query language**: $q(\overline{x}) \leftarrow A(x), r(x, y), U(x, z), \ldots, R_1(\overline{z}_1), \ldots, R_m(\overline{z}_m)$, where each $R(\overline{z}_i)$ is a datatype atom over $\mathcal{D}$; the red part is called the datatype pattern of $q$. 
Our main results are:

1. A framework for transferring classification results from constraint satisfaction problems (CSPs);
2. An instantiation of the datatype \((\mathbb{Q}, \leq)\), that with the use of a recent result gives us a classification of PTIME / NP-hard cases.
Framework (intuition behind theorem): given a query evaluation problem in DL-Lite over $\mathcal{D}$, we can translate it into a CSP over a similar structure $\mathcal{D}'$ (and vice-versa).

Therefore if we have a classification for CSPs, we can transfer it to query evaluation.
Classification for \((\mathbb{Q}, \leq)\)

Bodirsky and Kára\(^1\) recently obtained a deep classification result for temporal CSPs, that is, CSPs over \((\mathbb{Q}, R_1, R_2, \ldots)\) where each relation \(R_i\) is FO-definable on \((\mathbb{Q}, <)\).

Example of temporal CSP is the one over \(\mathcal{D} = (\mathbb{Q}, \text{Betw})\) where
\[
\text{Betw} = \{(a, b, c) \in \mathbb{Q}^3 \mid a < b < c \lor c < b < a\}.
\]

The classification is in terms of a certain algebraic property of the relations:

- If all \(R_i\) satisfy this property, then the CSP is in PTIME;
- Otherwise it is NP-complete.

\(^1\)The complexity of temporal CSPs, Journal of the ACM 2010.
**Results II: Instantiation of** \((Q, \leq)\)

\[ \mathcal{T} = \{ \text{Manager} \sqsubseteq \exists \text{salary}. \geq 40k, \ldots \} \]

\[ q(x) \leftarrow \text{Employee}(x), \text{Employee}(y), \text{salary}(x, v), \text{salary}(y, w), v \leq w \]

Query evaluation of \(q\) w.r.t. \(\mathcal{T}\)

PTIME

CSP over \((Q, R_T, R_q)\)

NP-hard

Reduction
Using our framework and Bodirsky and Kára’s result, using the same algebraic approach we showed a classification based on the datatype pattern \( q_0 = x_1 \leq y_1 \land \ldots \land x_n \leq y_n \).

For instance, given the datatype pattern \( q_0 = (x \leq y) \land (x \leq z) \):

\[
\begin{align*}
\text{PTIME} & \quad \text{for all } T, q \text{ where } q \supseteq q_0 \\
\text{co-NP-hard} & \quad \text{there exists } T \text{ and } q \text{ where } q \supseteq q_0
\end{align*}
\]
SYNTHESIS OF OUR CLASSIFICATION RESULT

We denote:

- **min-pattern**: $x_0 \leq x_1 \land x_0 \leq x_2 \land \ldots \land x_0 \leq x_n$
- **max-pattern**: $x_1 \leq x_0 \land x_1 \leq x_0 \land \ldots \land x_n \leq x_0$

Let $q_0$ be a datatype pattern over $(\mathbb{Q}, \leq)$. Then:

1. If $q_0$ is a max-pattern or a min-pattern, then evaluating $(\mathcal{T}, q)$, where $q \supseteq q_0$, is in PTIME in data complexity.
2. Otherwise there is an OMQ $(\mathcal{T}, q)$ with $q \supseteq q_0$ such that evaluating $(\mathcal{T}, q)$ is co-NP-complete in data complexity.
for all $\mathcal{T}, q$ where $q \supseteq q_0$

$q_0$ is a min- or a max-pattern

there exists $\mathcal{T}$ and $q$ where $q \supseteq q_0$
CONCLUSION AND FUTURE WORK

We created a bridge to CSPs and showed a classification result for the evaluation of OMQs for DL-Lite over $(\mathbb{Q}, \leq)$, providing a syntactical criterion for that purpose. Natural next steps are:

- Using our general framework and tools: transfer results for different datatypes.
- Developing practical query answering algorithms, in particular using constraint solvers as part of the query engines.