Complexity of query answering in lightweight Description Logics with datatypes
(thesis summary)

Julio Lemos
Department of Computer Science - University of Liverpool
jlemos@liverpool.ac.uk

1 Introduction and motivation

Description Logics (DLs) are a family of knowledge representation formalisms. In recent years they have been used for describing ontologies (Baader et al. 2003), for conceptual data modelling (Calvanese, Lenzerini, and Nardi 1998), the Semantic Web (Baader, Horrocks, and Sattler 2003) and ontology-based data access (OBDA) (Poggi et al. 2008), among others.

The search for tractable – and sufficiently expressive – fragments for the purposes just mentioned inspired the design of the so-called lightweight DLs, two of which are of note: EL (Baader, Brandt, and Lutz 2005) and DL-Lite (Calvanese et al. 2007). Both subfamilies stand at the basis of some of the W3C standard Web Ontology Language - OWL 2.0 profiles (Krötzsch 2012). The feature that interests me is the introduction of concrete domains, that is, datatypes and operations on them. It mainly consists in directly encoding in the logic values that correspond to datatypes used in applications: strings, integers, rational numbers, Boolean values etc. In the semantics this is effected by adding, alongside relations between objects, attributes, i.e., relations between objects and concrete (data) values whose interpretation is fixed.

In addition to traditional reasoning problems in DLs such as subsumption and satisfiability, and similarly to instance-checking (the task of checking whether some individual is an instance of a concept), recently the task of (conjunctive) query answering has been extensively investigated; see e.g. (Calvanese et al. 2007), (Glimm et al. 2011). In the OBDA framework, a DL ontology is used for specifying high-level notions (concepts and relations) of some domain representing knowledge that is also specified in a traditional database. Then it is possible to query the database without having to know how the data is represented at the level of the database: this is done by posing queries formulated in the language of the ontology.

Lightweight DLs with datatypes have been studied extensively in the last years (Magka, Kazakov, and Horrocks 2011), also in the context of OBDA (Artale, Ryzhikov, and Kontchakov 2012), (Savkovic and Calvanese 2012), (Franconi, Ibáñez-García, and Seylan 2011). It has been shown that query answering easily becomes intractable, so that looking for conditions under which it can be done in polynomial time is crucial. For queries that only contain unary predicates over data values (where e.g. we can retrieve employees in a company database whose salary is at least £50,000.00) such conditions have been proposed in (Savkovic and Calvanese 2012) and (Artale, Ryzhikov, and Kontchakov 2012) in the context of DL-Lite. These conditions consist in restrictions on the datatypes and their interaction.

Our proposal is to relax the restriction to unary datatypes, i.e., allowing queries containing predicates of higher arity over data values. In the example above, that would correspond e.g. to allowing comparisons between salaries of employees. On the other hand, since by allowing higher arity predicates over data values the query answering problem again becomes intractable in general, the goal is to carry out a non-uniform analysis of the complexity of query answering in the setting of lightweight languages—both the DL-Lite and EL families—with datatypes by taking into account the structure of both the ontology and the query.

2 Problem being addressed

Given the motivation above, we investigate the query answering problem in lightweight ontology languages parametrised with datatypes. At this stage of the project we focus on the core language of the DL-Lite family (Calvanese et al. 2007), nonetheless the plan is to extend the language later once a better understanding of this case is attained.

In order to understand the framework and the core problem, a formal definition of the main notions is needed.

As a rule we restrict ourselves to a single datatype modelled by a relational structure; and that is w.l.o.g., since several datatypes can be combined into a single structure. So let \( B = (\text{dom}(B); R_1, R_2, \ldots) \) be a relational structure where \( \text{dom}(B) \) is a set and each \( R_i \) is a \( k_i \)-ary relation on elements of \( \text{dom}(B) \). The elements of \( \text{dom}(B) \) are called data values.

We refer to such structures as datatypes. To keep the presentation simple, we focus on \( B := Q \) where \( Q := (\mathbb{Q}, \leq) \), that is, the rational numbers with their natural linear order. In general we are also interested in relational structures such as \( (\mathbb{Z}, <) \) and \( (\mathbb{R}, \neq) \), as well as in other logical structures with constants and functions.

We define a class of ontology languages parametrising DL-Lite\(_{\text{core}} \) with datatype \( B \): we denote it by DL-Lite\(_{\text{core}}(B) \).
A DL-Litecore\(_B\)-TBox consists of concept inclusions

\[ B \sqsubseteq C, \]

where \( B \) and \( C \) are defined according to the following grammar:

\[
\begin{align*}
B & ::= A \mid \exists p \mid \exists U, \\
C & ::= B \mid \neg B, \\
p & ::= r \mid r^-, 
\end{align*}
\]

where \( A \) ranges over concept names, \( r \) over role names and \( U \) over attribute names. Notice that as regards the language considered here, \( B \) does not affect the syntax, but only the semantics (i.e., values of attributes are from \( \text{dom}(B) \)). This will nonetheless change as soon as we extend the language.

A DL-Litecore\(_B\)-ABox is a set of assertions of the form \( A(a) \) (concept assertions), \( r(a,b) \) (role assertions) or \( U(a,v) \) (attribute assertions) where \( a, b \) are individual names and \( v \) a data value from \( \text{dom}(B) \). A pair consisting of a TBox and an ABox is here referred to as a knowledge base.

An interpretation \( I = (\Delta^I, \cdot) \) consists of a domain \( \Delta^I = \Delta_{\text{ind}}^I \cup \text{dom}(B) \), where \( \Delta_{\text{ind}}^I \) and \( \text{dom}(B) \) are disjoint, and an interpretation function \( \cdot \) that assigns to each individual name \( a \) an individual \( a^I \in \Delta_{\text{ind}}^I \) to each concept name \( A \) a set \( A^I \subseteq \Delta_{\text{ind}}^I \) to each role name \( r \) a set \( r^I \subseteq \Delta_{\text{ind}}^I \times \Delta_{\text{ind}}^I \) and to each attribute name \( U \) a set \( U^I \subseteq \Delta_{\text{ind}}^I \times \text{dom}(B) \). We also set \( v^I = v \in \text{dom}(B) \) for all data values \( v \in \text{dom}(B) \).

The reasoning task of query answering is defined in the following way.

A conjunctive query (CQ) is an expression \( q(\pi) \leftarrow \varphi(\pi, \bar{y}) \) where \( \varphi(\pi, \bar{y}) \) is a conjunction of atoms with variables from \( \pi \cup \bar{y} \), where each variable in \( \bar{x} \) occurs in some atom of \( \varphi \). The variables \( \pi \) are referred to as answer variables. The remaining variables are existentially quantified. A CQ without answer variables is called a Boolean conjunctive query.

Let \( (T, A) \) be a DL-Litecore\(_B\) knowledge base. In our setting, a CQ over the TBox \( T \) is a CQ whose atoms are of the form \( A(x) \), \( r(x,y) \), \( U(x,z) \) or \( R(\pi) \) where \( A \) is a concept name, \( r \) a role name, \( U \) an attribute name and \( R \) is a \( k \)-ary relation symbol from \( B \), where \( k = |\bar{z}| \), and \( x, y \) are variables ranging over individual names and \( z, \pi \) variables ranging over data values. The conjunction of all atoms of the form \( R(\bar{z}) \) in \( q \) is called the constraint part of the query, denoted \( q_c \). The conjunction of all remaining atoms is called the ontology part of the query, denoted \( q_o \). In short, we write \( q(\pi) \leftarrow q_o(\pi, \bar{y}), q_c(\pi) \). In this notation, \( \pi, \bar{y}, \bar{z} \) are answer variables, individual variables and data variables, respectively. Note that the answer variables are contained in the union of the sets of individual and data variables.

By \( \text{dom}(A) \) we denote all the individual names and elements of \( \text{dom}(B) \) that occur in \( A \). Now we define what it means for a query to be entailed by a knowledge base.

**Definition 2.1.** Let \( (T, A) \) be a DL-Litecore\(_B\) knowledge base, \( q(\pi) \leftarrow \varphi(\pi, \bar{y}) \) be a CQ and \( \bar{a} \in \text{dom}(A)^{|\bar{y}|} \). Then \( T, A \models q(\pi) \) iff for all models \( I \) of \( T \cup A \), there is an assignment \( \pi \) of the variables in \( \pi \cup \bar{y} \) into \( \Delta^I \) such that \( \pi(\pi) = \pi \) and \( I \models \varphi(\pi, \bar{y}) \).

In the context of the definition above, \( \pi \) is called an answer to \( q(\pi) \) in \( I \).

Given the definitions above, our main aim is to investigate the following problem, given a DL-Litecore\(_B\)-TBox \( T \) and a CQ \( q(\bar{x}) \) over \( \mathcal{T} \):

\[
\begin{array}{ll}
\text{Input:} & \text{a DL-Litecore\(_B\) ABox A, a tuple } \bar{a} \in \text{dom}(A)^{|\bar{x}|} \\
\text{Question:} & T, A \models q(\bar{a})? \\
\end{array}
\]

That is, we are interested in the data complexity\(^2\) of query answering in DL-Litecore\(_B\).

**Example 2.2.** To illustrate this problem with a very simple example, let

\[
A = \{ \text{City}(Leeds), \text{City}(Birmingham), \text{City}(Oxford), \\
\text{City}(Liverpool), \text{northing}(Leeds, 434260), \\
\text{northing}(Liverpool, 390631), \text{northing}(Oxford, 206186) \},
\]

\( T = \{ \text{City} \sqsubseteq \text{ Northing} \} \)

be a DL-Litecore\(_Q\) knowledge base. We formulate the following query over \( T, A \):

\[
q(x,y) \leftarrow \text{City}(x), \text{City}(y), \text{northing}(x, z_1), \text{northing}(y, z_2), \\
z_1 \leq z_2,
\]

which will give us all pairs of cities in \( A \) where the second is situated north of the first (pairing with “easting”, the “northing” value corresponds to the y-coordinate in the Cartesian plane). Notice that the information is incomplete, so that depending on the model, given some city, Birmingham might be or might not be situated north of it.

### 3 Proposed plan of research

After reviewing the literature and looking at possible applications where query answering with higher arity predicates is desirable, our aim is a non-uniform data complexity analysis of the problem \( QA(T, q, B) \). This analysis consists in looking at both syntactical restrictions on the query \( q \) and conditions on the TBox \( T \) that ensure tractability. That includes investigating conditions for first-order rewritability.

As a starting point we will look at \( QA(T, q, (Q, \leq)) \) where the ontology language is DL-Litecore\(_Q\)(\( Q, \leq \)).

Once this problem is well understood, we plan to (1) change the structure \( B \) by allowing ones with different numerical domains and relations as well as logical structures other than relational structures (e.g. structures with function symbols and/or constants), motivated by possible applications in the context of temporal and spatial reasoning; (2) extend the basic ontology language beyond DL-Litecore with datatypes, in addition to working with languages from the \( EL \) family; and possibly (3) introduce aggregates and non-monotonic negation in the query language, given their desirability in applications.

---

\(^2\)See (Vardi 1982).
4 Progress to date

It is not difficult to see that for many structures $B$, in particular $B = (\mathbb{Q}, \leq)$, we can construct a DL-Lite$_{core}(B)$-TBox $T$ and a CQ $q$ such that $QA(T, q, B)$ is coNP-hard. This motivated a non-uniform complexity analysis (see (Lutz and Wolter 2012)) of the problem, which was carried out by looking at sufficient conditions that ensure query answering in polynomial time. Ideally what we want is a classification result tied to syntactical restrictions on $q$.

Our first result establishes a very close link between the complexity of $QA(T, q, B)$ and the complexity of evaluating positive existential sentences in $B$, that is, the structure complementary to $B$.

A positive existential sentence is a first-order sentence built from atomic formulas by using solely conjunction, disjunction and existential quantifiers. Such a sentence is in Conjunctive Normal Form (CNF) if it has the form

$$\exists \bar{x} \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \varphi_{i,j},$$

where $\varphi_{i,j}$ are atomic formulas. If $n_i = k$, for each $i$, then we say that the sentence is in $k$-CNF.

So we proved the following proposition:

**Proposition 4.1.** Let $T$ be a DL-Lite$_{core}(B)$ TBox, and let $q(\bar{x}) \leftarrow q_0(\bar{y}, \bar{z}), q_c(\bar{z})$ be a CQ $q$ where $q_c$ has $k$ atoms, for $k \geq 1$. Assume either that (a) the chase of $T, A$ is finite for all $\bar{a}$, or that (b) $\bar{a}$ is connected and has at least one answer variable. Then we can compute in polynomial time for any ABox $A$ and tuple $\bar{a} \in dom(\bar{A})$ a positive existential sentence $\varphi$ in $k$-CNF that is true in $B$ iff $T, A \not|= \phi(\bar{a})$.

We strongly conjecture that this proposition holds even if the chase is not assumed to be finite.

In (Bodirsky and Kára 2010a) a dichotomy of the complexity of $QA(T, q, B)$ in terms of $\Gamma$ is proved. We use this result to obtain our dichotomy result as follows.

First we translate query answering over DL-Lite$_{core}(\mathbb{Q}, \leq)$ into the temporal CSP framework. So let $T, A$ be a DL-Lite$_{core}(\mathbb{Q}, \leq)$ knowledge base and $q$ be a Boolean CQ. For ease of exposition, assume that $q$ has a single component (i.e., all variables in the corresponding Gaifman graph are connected). It turns out that we can translate the problem of evaluating the positive existential sentence $\varphi$ obtained by Proposition 4.1 from $T, q$ into an instance $\psi$ of CSP($\mathbb{Q}, R_{qc}$), where $R_{qc}$ is a relation containing all the falsifying assignments to the variables in $q_c$.

In order to get $\psi$ we replace each conjunct $\bigwedge_{i=1}^{k} \varphi_{i,j}$ of $\varphi$ by a single atom of the form $R_{qc}(\cdot)$. Thus $T, A \not|= \psi$ if $\psi$ is a “yes” instance of CSP($\mathbb{Q}, R_{qc}$).

In fact, the classification result of (Bodirsky and Kára 2010a) consists in the identification of closure properties of the relations in $\Gamma$ that guarantee tractability. One of these closure properties corresponds, as pointed out in (Bodirsky and Kára 2010b), to a simple syntactical condition, called $l$-Horn or dual-$l$-Horn, on the formula defining a relation in $\Gamma$.

We have proved that restricting attention to relations of the form $R_{qc}$, the said $l$-Horn or dual-$l$-Horn condition corresponds to $R_{qc}$ being definable either by a formula of the form $x_1 < x_0 \lor x_2 < x_0 \lor \ldots \lor x_1 < x_0 (l$-Horn$)$ or by a formula of the form $x_1 > x_0 \lor x_2 > x_0 \lor \ldots \lor x_1 > x_0 (dual-l$-Horn$)$, Moreover, we proved that all other closure properties are equivalent to $l$-Horn/dual-$l$-Horn on relations of the form of $R_{qc}$.

This yields a simple syntactical classification of tractability for $QA(T, q, (\mathbb{Q}, \leq))$ based on the form of $q_c$. In particular, we obtain the desired dichotomy:

In general, the datatype could be any arbitrary dense linear order.

Formally, let $\bar{z} = (z_1, \ldots, z_k)$ denote all the data variables in $q_c$. Define a $k$-ary relation $R_{qc}$ on $\mathbb{Q}$ as follows: $R_{qc} := \{(\pi(z_1), \ldots, \pi(z_k)) \mid \pi : \{z_1, \ldots, z_k\} \rightarrow \mathbb{Q}, (\pi, \pi) \models \neg q_c\}$. 

---

3That is, $\bar{B}$ is the structure with $\text{dom}(\bar{B}) = \text{dom}(\bar{B})$ containing the relation $R := \text{dom}(\bar{B})^{\times k} \setminus R$ for each $k$-ary relation $R$ from $\bar{B}$ and all the functions and constants from $\bar{B}$.
Theorem 4.3. Let \( q_c(\overline{z}) \) be a conjunction of atoms over \((Q, \leq)\). If \( q_c(\overline{z}) \) is in \( ll\)-Horn or dual-\(ll\)-Horn form, then for every \( DL-Lite_{core}(Q, \leq)\)-TBox \( T \) and every Boolean CQ \( q \) over \( T \) whose constraint part is \( q_c \), \( QA(T, q, (Q, \leq)) \) can be solved in polynomial time. Otherwise, there is a \( DL-Lite_{core}(Q, \leq)\)-TBox \( T \) and a Boolean CQ \( q \) over \( T \) whose constraint part is \( q_c \) such that \( QA(T, q, (Q, \leq)) \) is \( \text{coNP-complete} \).

5 Related work

The problem described above—i.e., conjunctive query answering where queries include higher-arity predicates over data values—has not been considered in the past. Nonetheless this work builds on the investigation of:

1. the Description Logic \( EL \) extended with concrete domains and, more specifically, numerical domains, discussed in (Baader and Hanschke 1991), (Baader and Hanschke 1992), (Lutz 2002), (Magka, Kazakov, and Horrocks 2011);
2. \( DL-Lite \) parametrised with datatypes (mainly \( DL-Lite_\alpha \)), discussed in (Savkovic and Calvanese 2012), (Artale, Ryzhikov, and Kontchakov 2012), (Motik and Horrocks 2008);
3. the problem of query answering with \textit{unary} predicates over data values, fully discussed the context of \( DL-Lite_\alpha \) in (Poggi et al. 2008), (Calvanese et al. 2007) among others.

The main result we use is that query answering in \( DL-Lite_\alpha \)—which is an extension of \( DL-Lite_{core} \) with attributes over data values and which allow for axioms on datatypes as well as role and attribute functionality assertions—is \( \text{PTIME} \) in data complexity (cf. theorem 7 in (Poggi et al. 2008)); this only works provided that the cardinality of all used datatypes is infinite (a.k.a. “unbounded datatypes”), and that combining datatypes in such a way as to constrain individuals to take one among a finite number of attribute values is not allowed. That result is nonetheless not surprising, given that in query answering unary predicates over data values (“datatype assertions”) behave similarly to unary predicates over (an infinite number of) individuals, that is, concept assertions. As we have seen, introducing in the query language binary predicates over data values considerably complicates the problem.

References