What to Ask to an Incomplete Semantic Web Reasoner?

Bernardo Cuenca Grau and **Giorgos Stoilos**
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, UK
{berg,gios}@comlab.ox.ac.uk

Abstract

Largely motivated by Semantic Web applications, many highly scalable, but *incomplete*, query answering systems have been recently developed. Evaluating the scalability-completeness trade-off exhibited by such systems is an important requirement for many applications. In this paper, we address the problem of formally comparing complete and incomplete systems given an ontology schema (or TBox) $\mathcal{T}$. We formulate precise conditions on TBoxes $\mathcal{T}$ expressed in the EL, QL or RL profile of OWL 2 under which an incomplete system is indistinguishable from a complete one w.r.t. $\mathcal{T}$, regardless of the input query and data. Our results also allow us to quantify the "degree of incompleteness" of a given system w.r.t. $\mathcal{T}$ as well as to automatically identify concrete queries and data patterns for which the incomplete system will miss answers.

1 Introduction

Ontology schemas (or TBoxes) are often used for describing the meaning of data stored in various sources. In this setting, query languages are based on conjunctive queries (CQs), with the ontology providing the vocabulary used in queries [Glimm *et al.*, 2007; Lutz *et al.*, 2009; Calvanese *et al.*, 2007].

Largely motivated by Semantic Web applications, there has been a growing interest in the development of ontology-based query answering systems that are highly scalable in practice, but that are *incomplete*, i.e., they are not guaranteed to compute all query answers for some combinations of queries, ontologies, and datasets accepted as valid inputs. Examples of widely-used such systems are Oracle's Semantic Data Store, Jena, DLEJena, OWLim, Minerva and Sesame.

A challenge when using incomplete systems is to evaluate the scalability-completeness trade-off exhibited by different systems for a given application. As recently pointed out [Stoilos *et al.*, 2010b], the use of existing *performance* evaluation benchmarks, such as LUBM [Guo *et al.*, 2005], has serious limitations for this purpose. In particular, results using these benchmarks tell us little about the system's behaviour for the application at hand: they are limited to a specific ontology schema (the LUBM TBox about an academic domain), queries (LUBM contains a fixed pre-computed set of 14 sample queries), and datasets (a fixed number of relational structures hard-coded into the benchmark's scripts). Furthermore, correct answers can only be measured by comparison with the output of a complete system, and cannot be verified for large datasets that no complete system can handle.

The framework in [Stoilos *et al.*, 2010b; 2010a] partly addresses these limitations. The intuition behind these works is that given a TBox $\mathcal{T}$ and query q, it may be possible to compute a finite collection of datasets, called a *testing base*, such that if a system is complete for $\mathcal{T}$, q and each dataset in the collection, then it will also be complete for any dataset. In this case, one could say that an incomplete system is $(q, \mathcal{T})$-complete—that is, it behaves exactly like a complete reasoner w.r.t. q and $\mathcal{T}$, and regardless of the data (which is typically unknown and/or frequently changing). Furthermore, answers for each dataset in a testing base are known at generation time (thus, easy to verify), and the framework also provides a quantitative measure of completeness that makes comparison between different systems possible. Although there are ontologies and queries for which such testing base would necessarily be infinite, Stoilos et al. also identified sufficient conditions on $\mathcal{T}$ and q under which a (finite) testing base is guaranteed to exist and provided algorithms for computing it.

This framework, however, relies on the assumption that both the TBox and queries of interest are known. Although in many applications the TBox is under the control of the application developers, queries depend on users' needs and application developers may only have a rough idea of which ones might be of particular relevance. Consequently, the assumption that test queries should be fixed in advance for evaluation purposes is likely to be too strict a requirement.

In this paper, we address this limitation and develop a novel framework in which completeness evaluation depends *only* on the application's TBox $\mathcal{T}$, and not on a set of pre-defined queries. More precisely, given $\mathcal{T}$, we study the problem of generating a finite collection of conjunctive queries (a *query testing base*), such that if a system is $(q, \mathcal{T})$-complete for each query q in the collection, then it will also be $(q', \mathcal{T})$-complete for *any* arbitrary query q'. As well as providing a new quantitative measure of completeness depending only on $\mathcal{T}$, this would allow us to identify circumstances under which an incomplete reasoner is indistinguishable from a complete one for a particular TBox $\mathcal{T}$, regardless of the user's query and input data. As a result, application developers would be able to assess with a much higher degree of confidence whether a given system meets their needs.

In practice, our framework relies on the techniques developed in [Stoilos *et al.*, 2010b] and [Stoilos *et al.*, 2010a] in order to check whether a system is $(q, \mathcal{T})$-complete for each q in a query testing base (and hence we are subject to their same practical limitations). Conceptually, however, we are addressing a very different problem since we are interested in automated *query generation*, rather than in data generation.

It is worth emphasising that automated query generation is an active research topic in databases [Poess and Stephens, 2004; Khalek and Khurshid, 2010]. The focus in the database literature, however, is on performance evaluation rather than on completeness of the evaluated query engines.

The main contributions of this paper are as follows:

- We extend the framework in [Stoilos *et al.*, 2010b] with the notion of a query testing base.
- We explore the limitations of the new framework and formulate precise conditions under which a query testing base would necessarily be infinite.
- We identify sufficient conditions on $\mathcal{T}$ for which a (finite) query testing base can be efficiently obtained. Interestingly, such sufficient conditions are strongly related to the notion of *weak acyclicity*, which is widely used in the context of data exchange [Fagin *et al.*, 2005].
- We present query generation algorithms for the logics underpinning the QL, EL and RL profiles of the new standard ontology language OWL 2 [Motik *et al.*, 2009].
- We provide preliminary empirical evidence suggesting that our framework is feasible in practice.

This paper is accompanied with an online technical report containing complete proofs and additional technical details.[1]

2 Preliminaries

Description Logics We assume basic familiarity with DL syntax, semantics and standard reasoning problems.

When in Section 3 we speak of an arbitrary description logic $\mathcal{L}$, we refer to a fragment of the DLs underpinning OWL and OWL 2. We use standard notions of an $\mathcal{L}$-TBox $\mathcal{T}$ (the schema), an $\mathcal{L}$-ABox $\mathcal{A}$ (the data), and an $\mathcal{L}$-ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$, but assume that ABoxes contain only atomic assertions of the form $A(a)$ or $R(a, b)$, with A and R atomic. Also, we sometimes use the standard notion of homomorphism (isomorphism) between ABoxes. Finally, to avoid conflating schema and data, we only consider DLs without nominals (i.e., that do not allow individuals in the TBox).

We next recapitulate the syntax of the concrete DLs mentioned in Section 4, namely DL-Lite[2] [Calvanese *et al.*, 2007] and $\mathcal{EL}$ [Baader *et al.*, 2005], which provide the logical underpinning for the QL and EL profiles of OWL 2.

A role is an atomic role R or its inverse R^-. The function $\mathsf{ar}(R, a, b)$ takes a role R and individuals a, b, and returns $R(a, b)$ if R is atomic, or $P(b, a)$ if $R = P^-$, with P atomic. A DL-Lite-concept B is either atomic, or of the form $\exists R.\top$ with R a role. A DL-Lite-TBox is a finite set of GCIs of the form $B_1 \sqsubseteq B_2$ (positive GCI), or $B_1 \sqsubseteq \neg B_2$ (negative GCI), with B_1 and B_2 DL-Lite concepts.

The set of $\mathcal{EL}$-concepts is given by the following grammar, where A, R are atomic and $C_{(i)}$ are $\mathcal{EL}$-concepts:

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists R.C$$

An $\mathcal{EL}$-TBox $\mathcal{T}$ is a finite set of GCIs of the form $C_1 \sqsubseteq C_2$ with C_1 and C_2 $\mathcal{EL}$-concepts. We use the notation $C_1 \equiv C_2$ as an abbreviation for $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. We assume from now on that $\mathcal{EL}$-TBoxes are normalised, i.e., that each GCI in $\mathcal{T}$ is in one of the following forms, with $A_{(i)}$ atomic or $\top$, and R atomic: $A_1 \sqsubseteq A_2$, $A_1 \sqcap A_2 \sqsubseteq A$, $A_1 \sqsubseteq \exists R.A_2$, or $\exists R.A_2 \sqsubseteq A_1$.

The DLs considered in this paper can be seen as fragments of First-Order Logic (FOL), and their semantics is standard [Baader *et al.*, 2002]. The definitions of reasoning problems such as consistency and entailment are also standard.

Conjunctive Queries We use standard notions of (function-free) term and variable. A concept atom is of the form $A(t)$ with A an atomic concept and t a term. A role atom is of the form $R(t, t')$ for R an atomic role, and t, t' terms. A conjunctive query (CQ) q is an expression of the form

$$q(x_1, \ldots, x_n) \leftarrow \{\alpha_1, \ldots, \alpha_m\}$$

where each α_i is a concept atom or a role atom and each x_j is a *distinguished* variable occurring in some α_i. A certain answer to q w.r.t. $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is a tuple $\mathbf{c} = \langle c_1, \ldots, c_n \rangle$ of individuals such that $\mathcal{O}$ entails the FOL formula obtained by building the conjunction of all atoms α_i in q, replacing each distinguished variable x_j with c_j and existentially quantifying over the remaining variables. We denote with $\mathsf{cert}(q, \mathcal{O})$ the set of all certain answers to q w.r.t. $\mathcal{O}$. Finally, we sometimes use the well-known notions of homomorphism from a query q to q' and between a CQ and an ABox.

The Chase CQ answering in $\mathcal{L} \in \{\text{DL-Lite}, \mathcal{EL}\}$ can be characterised according to the notion of *chase*, adapted from database dependency theory. Given a consistent $\mathcal{L}$-ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$, $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ is a (possibly infinite) forest-shaped ABox constructed step-by-step from $\mathcal{A}$ and which represents all models of $\mathcal{O}$ for the purpose of CQ answering. More precisely, $\mathsf{cert}(q, \mathcal{O}) = \mathsf{cert}(q, \mathsf{chase}_{\mathcal{L}}(\mathcal{O}))$, where we abuse notation and use $\mathsf{cert}(q, \mathsf{chase}_{\mathcal{L}}(\mathcal{O}))$ to represent those answers to q w.r.t. $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ containing only individuals from $\mathcal{A}$ (which we call *named* chase individuals from now on).

The chase construction rules for DL-Lite and $\mathcal{EL}$ are given in [Calvanese *et al.*, 2007] and [Rosati, 2007], respectively. They are also given in a compact way in our technical report.

3 Framework

We next present our query generation framework. We start by recalling the notion of a CQ answering algorithm given in [Stoilos *et al.*, 2010b], which will allow us, on the one hand, to abstract from the specifics of implemented systems and, on the other hand, to establish general results that hold for any system satisfying certain basic properties.

[1]http://ijcai2011.tripod.com/querycompleteness.pdf

[2]The logic used in this paper is the simplest one among those in the DL-Lite family, and it is commonly referred to as DL-Lite$_{core}$.

Definition 1. A *CQ answering algorithm* ans for a DL $\mathcal{L}$ is a procedure that, for each $\mathcal{L}$-ontology $\mathcal{O}$ and CQ q computes in a finite number of steps a set of tuples $\mathsf{ans}(q, \mathcal{O})$ of the same arity as the certain answers to q.

It is *sound* if, for each $\mathcal{O}$ and q, $\mathsf{ans}(q, \mathcal{O}) \subseteq \mathsf{cert}(q, \mathcal{O})$, and it is *complete* if $\mathsf{cert}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O})$.

It is *monotonic* if $\mathsf{ans}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O}')$ for each q, $\mathcal{O}$ and $\mathcal{O}'$ such that $\mathcal{O} \subseteq \mathcal{O}'$.

It is *invariant under renamings* if, for each q, $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$, and $\mathcal{O}' = \mathcal{T} \cup \mathcal{A}'$ with $\mathcal{A}$ and $\mathcal{A}'$ isomorphic, $\mathsf{ans}(q, \mathcal{O})$ and $\mathsf{ans}(q, \mathcal{O}')$ are identical modulo the same isomorphism.

Finally, ans is *well-behaved* if it is sound, monotonic and invariant under renamings.

Intuitively, a well-behaved algorithm implements the semantics of CQ answering faithfully: query answers can only grow if we add axioms to the ontology, and they cannot depend on trivial renamings of ABox individuals. This allows us to rule out "unreasonable" algorithms at this level of generality. All incomplete systems known to us are well-behaved.

We next recall the notion of completeness for a fixed query q and TBox $\mathcal{T}$ studied in [Stoilos *et al.*, 2010b; 2010a].

Definition 2. Let $\mathcal{L}$ be a description logic, q a CQ, $\mathcal{T}$ an $\mathcal{L}$-TBox and ans a CQ answering algorithm for $\mathcal{L}$. We say that ans is $(q, \mathcal{T})$*-complete* if for each ABox $\mathcal{A}$ s.t. $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is consistent, we have that $\mathsf{cert}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O})$.

We can then introduce our notion of a *query testing base*: a set of CQs for a TBox $\mathcal{T}$ that we can use to ensure that an incomplete reasoner behaves exactly like a complete one for $\mathcal{T}$, regardless of the input query and data.

Definition 3. Let $\mathcal{L}$ be a description logic, and let $\mathcal{T}$ be an $\mathcal{L}$-TBox. A *Query Testing Base* (QTB) $\mathbf{Q}$ for $\mathcal{T}$ and a class $\mathcal{C}^{\mathcal{L}}$ of CQ answering algorithms for $\mathcal{L}$ is a finite set of CQs such that the following property holds for each algorithm ans in $\mathcal{C}^{\mathcal{L}}$: If ans is $(q, \mathcal{T})$-complete for each $q \in \mathbf{Q}$, then it is also $(q', \mathcal{T})$-complete for each CQ q'.

To make our results as general as possible, we have parameterised QTBs w.r.t. a given class of algorithms—that is, a family of algorithms that share certain properties.

Unfortunately, as shown by the following theorem, QTBs fail to exist even for the empty ontology because such a QTB would need to contain infinitely many queries.

Theorem 4. *Let $\mathcal{L}$ be a DL. No QTB exists for $\mathcal{T} = \emptyset$ and the class of well-behaved CQ answering algorithms for $\mathcal{L}$.*

Proof. Let $\mathbf{Q}$ be an arbitrary, but finite, set of CQs and let m be the maximum number of variables in a query from $\mathbf{Q}$. Consider also the following query:

$$q(x) \leftarrow \{R(x, y_1), R(y_1, y_2), \ldots, R(y_{m-1}, y_m)\}$$

We provide a well-behaved algorithm ans for $\mathcal{L}$ that is $(p, \emptyset)$-complete for each $p \in \mathbf{Q}$, but it is not $(q, \emptyset)$-complete, which proves our theorem. Let ans proceed as follows when given a CQ q_{in} and an $\mathcal{L}$-ontology $\mathcal{O}_{in} = \mathcal{T}_{in} \cup \mathcal{A}_{in}$:

- If $\mathcal{T}_{in} \neq \emptyset$, return $\mathsf{cert}(q_{in}, \mathcal{A}_{in})$
- If $\mathcal{T}_{in} = \emptyset$, do the following:
  1. If q_{in} has at most m vars., return $\mathsf{cert}(q_{in}, \mathcal{A}_{in})$.
  2. Otherwise, return the empty set.

The algorithm is clearly $(p, \emptyset)$-complete for each $p \in \mathbf{Q}$, as it returns $\mathsf{cert}(p, \mathcal{A}_{in})$. It is, however, incomplete for q, which contains $m+1$ variables, as it returns the empty set regardless of $\mathcal{A}_{in}$. Finally, ans is clearly sound, invariant under renamings (the way it operates does not depend on $\mathcal{A}_{in}$, but rather on the shape of q_{in} and $\mathcal{T}_{in}$), and also monotonic. □

The proof of Theorem 4 suggests that the notion of a well-behaved algorithm is too general: no matter how large a QTB is, we can find a (rather unnatural) well-behaved algorithm that is complete for all queries in the QTB, but incomplete for some other queries. To obtain useful results, we thus need to require additional properties to CQ answering algorithms.

Towards this goal, we observe that most incomplete systems are based on database or RDF triple store technologies. Given $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ and q as input, these systems first deterministically "saturate" $\mathcal{A}$ with new assertions using the knowledge in $\mathcal{T}$ and then answer the query q directly w.r.t. the saturated ABox, as if it was a database. Hence, their behaviour can be characterised at a general level as given next.

Definition 5. An *ABox-saturation* algorithm for $\mathcal{L}$ is an algorithm that given as input an $\mathcal{L}$-ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ and a CQ q proceeds as follows:

1. It computes a *saturation ABox* $\mathcal{A}_f$ whose contents depend only on $\mathcal{O}$, and such that $\mathcal{A} \subseteq \mathcal{A}_f$.
2. It returns those answers to q w.r.t. $\mathcal{A}_f$ containing only individuals from $\mathcal{A}$. From now on, we abuse notation and write $\mathsf{cert}(q, \mathcal{A}_f)$ to represent (only) such answers.

Furthermore, we assume that ans is *stable*: if $\mathcal{A}_f$ is the saturation ABox for $\mathcal{T} \cup \mathcal{A}$, then the saturation ABox for $\mathcal{T} \cup \mathcal{A}_f$ is also $\mathcal{A}_f$.

Note that the negative result in Theorem 4 does not apply to the class of well-behaved ABox-saturation algorithms. Indeed, ABox-saturation algorithms are always complete if the TBox is empty. If one such algorithm is incomplete, it is because it fails to capture relevant knowledge from the TBox.

As shown in the following theorem, however, there exist TBoxes expressed in rather simple ontology languages for which no QTB exists, even if we restrict ourselves to well-behaved ABox-saturation algorithms.

Theorem 6. *There is a TBox $\mathcal{T}$ in $\mathcal{L} \in \{\mathcal{EL}, \textit{DL-Lite}\}$ such that no QTB exists for the class of all well-behaved ABox-saturation algorithms for $\mathcal{L}$.*

Proof. (Sketch) Let $\mathbf{Q}$ be an arbitrary, but finite, set of CQs and let m be the maximum number of variables in a query from $\mathbf{Q}$. Consider the following $\mathcal{EL}$-TBox

$$\mathcal{T}_{el} = \{A \sqsubseteq \exists R.A\}$$

and the following query.

$$q(x) \leftarrow \{R(x, y_1), R(y_1, y_2), \ldots, R(y_{m-1}, y_m), A(y_m)\}$$

Similarly to the proof of Theorem 4, we can provide a well-behaved ABox-saturation algorithm that is $(p, \mathcal{T}_{el})$-complete for each $p \in \mathbf{Q}$, but not $(q, \mathcal{T}_{el})$-complete. Details are provided in our technical report. The proof for DL-Lite is analogous, by considering query q and the TBox $\mathcal{T}_{lite} = \{A \sqsubseteq \exists R.\top, \exists R^{-}.\top \sqsubseteq A\}$, which entails $\mathcal{T}_{el}$. □

4 Computing a Query Testing Base

In this section we identify fairly general, yet practical, sufficient conditions on TBoxes that guarantee the existence of a QTB and yield algorithms to compute such a QTB efficiently.

Towards this goal, we observe that the proof of Theorem 6 exploits the fact that we can have DL TBoxes which imply cycles over existential quantifiers. As a result, no matter how large the maximum role depth of a query in the QTB is, we can always define an algorithm that "unfolds" the cycle only up to that depth. Such algorithm may then be incomplete for queries with a larger role depth. Our intuition is that a QTB for $\mathcal{T}$ may be guaranteed to exist if we can ensure that such cycles do not occur (even implicitly) in $\mathcal{T}$.

In Section 4.1 we focus on the logics DL-Lite and $\mathcal{EL}$ and provide a sufficient condition for preventing cycles. We assume from now onwards that $\mathcal{EL}$ TBoxes are normalised as described in the preliminaries.

In Section 4.2 we focus on the description logics underpinning the RL profile of OWL 2 [Motik *et al.*, 2009], and show that for such DLs a QTB always exists.

4.1 DL-Lite and $\mathcal{EL}$

To preclude cycles in DL-Lite and $\mathcal{EL}$ TBoxes, we rely on the *weak acyclicity* condition borrowed from database theory [Fagin *et al.*, 2005] and which can be checked efficiently.

Roughly speaking, if $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ and $\mathcal{T}$ is weakly acyclic, then a run of the corresponding chase does not lead to an infinite generation of new individuals. Thus, weak acyclicity also prevents cyclic existential quantification.

The following definition is a straightforward application of the general notion of weak acyclicity to DL-Lite and $\mathcal{EL}$.

Definition 7. Let $\mathcal{T}$ be an $\mathcal{L}$-TBox, with $\mathcal{L} \in \{\text{DL-Lite}, \mathcal{EL}\}$. The *dependency graph* for $\mathcal{T}$ is the smallest graph with the following elements:

- *Nodes:* A node v_A for each atomic concept A and nodes $v_{R/1}$ and $v_{R/2}$ for each atomic role in $\mathcal{T}$.
- *Edges:* For R atomic and A, B either atomic or $\top$, let $[A] = A$, $[\exists R.B] = R/1$ and $[\exists R^{-}.B] = R/2$.
  - If $\mathcal{T} \in$ DL-Lite, then the graph contains the following edges for each positive GCI $(B \sqsubseteq C) \in \mathcal{T}$:
    - ∗ An edge $v_{[B]} \rightarrow v_{[C]}$.
    - ∗ A special edge $v_{[B]} \overset{*}{\rightarrow} v_{R/2}$ if $C = \exists R.\top$.
    - ∗ A special edge $v_{[B]} \overset{*}{\rightarrow} v_{R/1}$ if $C = \exists R^{-}.\top$.
  - If $\mathcal{T} \in \mathcal{EL}$, it has edges $v_A \rightarrow v_\top$ and $v_{R/1} \rightarrow v_\top$ for each atomic A and R, as well as the following edges for each $(\bigsqcap_{1 \leq i \leq n} B_i \sqsubseteq C) \in \mathcal{T}$ with $n \leq 2$:
    - ∗ Edges $v_{[B_i]} \rightarrow v_{[C]}$.
    - ∗ Special edges $v_{[B_i]} \overset{*}{\rightarrow} v_{[D]}$ and $v_{[B_i]} \overset{*}{\rightarrow} v_{[R/2]}$ if $C = \exists R.D$.

The dependency graph for $\mathcal{T}$ is *weakly acyclic* if it contains no cycle going through a special edge.

Intuitively, special edges signal the possible creation of a new individual via existential quantification during the chase construction, whereas the other edges keep track of possible propagation of information between existing individuals.

Example 8. Consider the following $\mathcal{EL}$-TBox:

$$\begin{aligned}\mathcal{T} \;=\; &\{\mathsf{Student} \equiv \exists \mathsf{takesCourse}.\mathsf{Course},\\ &\mathsf{GradCourse} \sqsubseteq \mathsf{Course},\\ &\mathsf{GradStudent} \sqsubseteq \exists \mathsf{takesCourse}.\mathsf{GradCourse}\}\end{aligned}$$

Although the dependency graph for $\mathcal{T}$ is cyclic, there is no cycle involving a special edge; hence, $\mathcal{T}$ is weakly acyclic.

The next theorem follows from a general result for weakly acyclic tuple-generating dependencies [Fagin *et al.*, 2005].

Theorem 9. *Let $\mathcal{L} \in \{\text{DL-Lite}, \mathcal{EL}\}$ and let $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ be a consistent $\mathcal{L}$-ontology such that $\mathcal{T}$ is weakly acyclic. Then, there is a polynomial in the size of $\mathcal{A}$ bounding the length of every chase sequence.*[3]

Hence, weak acyclicity establishes a bound on the size of the chase (and hence also of the canonical model of $\mathcal{O}$).

Since the chase is forest-shaped, each unnamed individual is connected to a named individual via a unique "path" of role assertions. Therefore, we can characterise the assertions generated by the application of chase rules as follows.

Definition 10. Let $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ be a consistent $\mathcal{L}$-ontology for $\mathcal{L} \in \{\text{DL-Lite}, \mathcal{EL}\}$. A *path* of $\mathsf{chase}_\mathcal{L}(\mathcal{O})$ is a subset of $\mathsf{chase}_\mathcal{L}(\mathcal{O})$ of the form $\{A(a)\}$ with A atomic, a named and $A(a) \notin \mathcal{A}$, or $\{\mathsf{ar}(R_1, a, b_1), \ldots, \mathsf{ar}(R_n, b_{n-1}, b_n), B(b_n)\}$ with B either atomic or $\top$, a named and each b_i unnamed.

We can then define a family of queries for checking whether an incomplete system captures the relevant information that could possibly be introduced by the application of chase rules. Such queries should depend only on the shape of the TBox $\mathcal{T}$ (and not on the particular input ABox).

Definition 11. Let $\mathcal{L} \in \{\text{DL-Lite}, \mathcal{EL}\}$, let $\mathcal{T}$ be an $\mathcal{L}$-TBox with $\mathcal{T}_p \subseteq \mathcal{T}$ the subset of $\mathcal{T}$ containing no negative GCIs, and let $\mathsf{LHS} = \{C \mid C \sqsubseteq D \in \mathcal{T}_p\}$. For each $C \in \mathsf{LHS}$, let c_1 and c_2 be individuals uniquely associated to C, and let

$$\mathcal{A}_C = \begin{cases} \{A(c_1)\} & \text{if } C = A \\ \{A_1(c_1), A_2(c_1)\} & \text{if } C = A_1 \sqcap A_2 \\ \{\mathsf{ar}(R, c_1, c_2), A(c_2)\} & \text{if } C = \exists R.A \end{cases}$$

where A is either atomic or $\top$. Let $\mathcal{A}_\mathcal{T} = \bigcup_{C \in \mathsf{LHS}} \mathcal{A}_C$. A CQ q is a *chase query* w.r.t. $\mathcal{T}$ if it can be obtained from a path of $\mathsf{chase}_\mathcal{L}(\mathcal{T} \cup \mathcal{A}_\mathcal{T})$ by replacing each individual c with a variable x_c such that x_c is distinguished iff c is named.[4]

Example 12. The following are all the chase queries (modulo isomorphisms) w.r.t. TBox $\mathcal{T}$ from Example 8:

$$\begin{aligned} q_1(x) &\leftarrow \{\mathsf{Course}(x)\}\\ q_2(x) &\leftarrow \{\mathsf{Student}(x)\}\\ q_3(x) &\leftarrow \{\mathsf{takesCourse}(x, y)\}\\ q_4(x) &\leftarrow \{\mathsf{takesCourse}(x, y), \mathsf{Course}(y)\}\\ q_5(x) &\leftarrow \{\mathsf{takesCourse}(x, y), \mathsf{GradCourse}(y)\}\end{aligned}$$

The following lemma shows that chase queries can be used to check for the presence of any relevant piece of new information introduced by the application of chase rules.[5]

[3]A chase sequence is any sequence of chase rule applications transforming $\mathcal{A}$ into $\mathsf{chase}_\mathcal{L}(\mathcal{O})$.

[4]Note that a chase query has exactly one distinguished variable.

[5]A detailed proof can be found in our technical report.

Lemma 13. *Let $\mathcal{L} \in \{$DL-Lite$, \mathcal{EL}\}$ and let $\mathcal{O}$ be a consistent $\mathcal{L}$-ontology. The CQ obtained by replacing in a path of $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ each (un)named individual c with an (un)distinguished variable x_c is a chase query w.r.t. $\mathcal{T}$.*

As a result, if an ABox-saturation algorithm is complete for each chase query, it should be possible to homomorphically embed $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ into the saturation ABox $\mathcal{A}_f$ computed by the algorithm for $\mathcal{O}$.[5]

Lemma 14. *Let $\mathcal{L} \in \{$DL-Lite$, \mathcal{EL}\}$ and let $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ be a consistent $\mathcal{L}$-ontology. Let ans be a well-behaved ABox-saturation algorithm that is $(q, \mathcal{T})$-complete for each chase query q for $\mathcal{T}$. Finally, let $\mathcal{A}_f$ be the saturation ABox computed by ans for $\mathcal{O}$. Then, there is a homomorphism μ from $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ to $\mathcal{A}_f$ that maps named individuals to themselves.*

Furthermore, weak acyclicity ensures the existence of a polynomial bound in both the size and number of chase queries for each input, as shown by the following Lemma.

Lemma 15. *Let $\mathcal{L} \in \{$DL-Lite$, \mathcal{EL}\}$. Each chase query for a weakly acyclic $\mathcal{L}$-TBox $\mathcal{T}$ is of size polynomial in the size of $\mathcal{T}$. Also, there are polynomially many non-isomorphic chase queries in the size of $\mathcal{T}$.*

Proof. (Sketch) Given $\mathcal{T}$ and an ABox $\mathcal{A}$, Theorem 9 implies that the the size of $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ is polynomial in the size of $\mathcal{A}$. Let $\mathcal{O}' = \mathcal{T} \cup \mathcal{A}_{\mathcal{T}}$ as in Definition 11. Since the size of $\mathcal{A}_{\mathcal{T}}$ is bounded by the size of $\mathcal{T}$, the size of $\mathsf{chase}_{\mathcal{L}}(\mathcal{O}')$ (and hence the size of each chase query) is polynomially bounded by the size of $\mathcal{T}$. Finally, the number of chase queries is bounded by the number of assertions in $\mathsf{chase}_{\mathcal{L}}(\mathcal{O}')$ and it is thus polynomially bounded by the size of $\mathcal{T}$.[5] □

Lemmas 14 and 15 suggest that the set of chase queries for a weakly acyclic TBox $\mathcal{T}$ constitutes a QTB for $\mathcal{T}$.

Theorem 16. *Let $\mathcal{L} \in \{$DL-Lite$, \mathcal{EL}\}$ and let $\mathcal{T}$ be a weakly acyclic $\mathcal{L}$-TBox. Then, the set of all chase queries (unique up to isomorphism) is a QTB for $\mathcal{T}$ and the class of well-behaved ABox-saturation algorithms for $\mathcal{L}$. Furthermore, the size of such QTB is polynomial in the size of $\mathcal{T}$.*

Proof. Let q be a CQ and let $\mathcal{A}$ be an ABox s.t. $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is consistent. Then, $\mathsf{cert}(q, \mathcal{O}) = \mathsf{cert}(q, \mathsf{chase}_{\mathcal{L}}(\mathcal{O}))$.

If $\mathbf{c} \in \mathsf{cert}(q, \mathsf{chase}_{\mathcal{L}}(\mathcal{O}))$, there is a homomorphism σ from q to $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ which maps the distinguished variables to $\mathbf{c}$. Since ans is $(q', \mathcal{T})$-complete for all chase queries q', Lemma 14 implies that there also exists a homomorphism μ between $\mathsf{chase}_{\mathcal{L}}(\mathcal{O})$ and $\mathcal{A}_f$ that maps named individuals to themselves. But then, σ composed with μ is a homomorphism from the variables of q to the individuals in $\mathcal{A}_f$ which maps distinguished variables to $\mathbf{c}$; hence, $\mathbf{c} \in \mathsf{cert}(q, \mathcal{A}_f)$. Since $\mathsf{ans}(q, \mathcal{O}) = \mathsf{cert}(q, \mathcal{A}_f)$ we have that $\mathbf{c} \in \mathsf{ans}(q, \mathcal{O})$, which implies that the set of all chase queries is a QTB. Finally, since $\mathcal{T}$ is weakly acyclic, Lemma 15 then implies that such QTB is of size polynomial in the size of $\mathcal{T}$. □

4.2 DLP

Description Logics Programs (DLP) [Grosof *et al.*, 2003] is a prominent family of DLs which provide the logical underpinning for the RL profile of OWL 2 [Motik *et al.*, 2009]. The logics in this family impose syntactic restrictions to avoid, on the one hand, the need to infer the existence of individuals not explicitly present in the ABox and, on the other hand, the need for nondeterministic reasoning. Such DLs have recently become popular since they are amenable to implementation using rule-based reasoning engines. In fact, many incomplete ontology reasoners use rule-based forward-chaining techniques to perform reasoning.

Therefore, DLP logics have been designed so as to satisfy the property given in the following lemma.

Lemma 17. *Let $\mathcal{L}$ be in DLP. For each $\mathcal{L}$-ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$, there is an ABox $\mathcal{A}'$ mentioning only individuals from $\mathcal{A}$ and such that $\mathsf{cert}(q, \mathcal{O}) = \mathsf{cert}(q, \mathcal{A}')$ for every CQ q.*

Since reasoning in DLP does not require the generation of "fresh" individuals, it is intuitive to expect that a QTB is guaranteed to exist. Furthermore, such QTB would only need to include queries asking for all instances of each atomic concept and each atomic role occurring in the ontology.

Theorem 18. *Let $\mathcal{L}$ be in DLP and let $\mathcal{T}$ be a $\mathcal{L}$-TBox. Let $\mathbf{Q}$ consist of a query $q_A(x) \leftarrow A(x)$ for each atomic concept A in $\mathcal{T}$ and a query $q_R(x, y) \leftarrow \mathsf{ar}(R, x, y)$ for each (possibly inverse) role occurring in $\mathcal{T}$. Then, $\mathbf{Q}$ is a QTB for $\mathcal{T}$ and the class of all well-behaved ABox-saturation algorithms.*

Proof. Let ans be a well-behaved ABox-saturation algorithm that is $(q', \mathcal{T})$-complete for each $q' \in \mathbf{Q}$. Let q be a CQ and let $\mathcal{A}$ be an ABox such that $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is consistent. We show that $\mathsf{cert}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O})$.

If $\mathbf{c} \in \mathsf{cert}(q, \mathcal{O})$, then $\mathbf{c} \in \mathsf{cert}(q, \mathcal{A}')$, for some $\mathcal{A}'$ as in Lemma 17. Hence, there is a homomorphism μ from q to $\mathcal{A}'$ mapping distinguished variables to $\mathbf{c}$ and s.t. for each body atom $A(t)$ (respectively $R(t, t')$) in q, $A(\mu(t)) \in \mathcal{A}'$ (respectively $R(\mu(t), \mu(t')) \in \mathcal{A}'$); furthermore, $\mu(t) \in \mathsf{ans}(q_A, \mathcal{O})$ and $\langle \mu(t), \mu(t') \rangle \in \mathsf{ans}(q_R, \mathcal{O})$ since $q_A, q_R \in \mathbf{Q}$ and ans is complete for q_A and q_R. But then, $\mu(t) \in \mathsf{cert}(q_A, \mathcal{A}_f)$ and $\langle \mu(t), \mu(t') \rangle \in \mathsf{cert}(q_R, \mathcal{A}_f)$, with $\mathcal{A}_f$ the saturation ABox for $\mathcal{O}$ computed by ans; thus, $A(\mu(t)) \in \mathcal{A}_f$ (respectively $R(\mu(t), \mu(t')) \in \mathcal{A}_f$), and $\mathbf{c} \in \mathsf{ans}(q, \mathcal{O})$, as required. □

5 Evaluation

We have implemented a prototype query generator for DL-Lite and computed a QTB for a DL-Lite version of the LUBM TBox. The computation of LUBM's QTB required less than a second and contained only 16 queries, denoted as $Q1$-$Q16$. (Note that the QTB is so small because many concepts never appear on the right-hand side of a GCI).

We have used this QTB to evaluate the following systems: Sesame 2.3-prl,[6] OWLim,[7] Jena v2.6.4[8] using both its OntModel and InfModel interfaces, DLEJena,[9] Oracle's Semantic Data Store[10] using both the RDFS and OWLPrime pro-

[6]http://www.openrdf.org/

[7]http://www.ontotext.com/owlim/

[8]http://jena.sourceforge.net/

[9]http://lpis.csd.auth.gr/systems/DLEJena/

[10]http://www.oracle.com/technetwork/database/options/semantic-tech/index.html

Sesame, Oracle RDFS						**DLEJena**		
Q8	Q9	Q11	Q14	Q15	Q16	Q14	Q15	Q16
.84	.89	.87	.5	.25	.15	.5	.25	.15

OWLim, Jena_O Micro, Oracle OWLPrime			
Q11	Q14	Q15	Q16
.87	.5	.25	.15

Table 1: Completeness degree for queries in LUBM's QTB

files, and Pellet v2.2.2.[11] To this end, we generated the required datasets for each query in the QTB using the techniques from [Stoilos *et al.*, 2010a], which we then used to compute the corresponding completeness degree for each system. (For DL-Lite TBoxes, a finite collection of such datasets is guaranteed to exist for each query [Stoilos *et al.*, 2010a]).

As expected, Pellet (a fully-fledged OWL 2 reasoner) was found to be complete for all queries. More interesting is the fact that Jena Full in its InfModel implementation was also complete for all queries and hence we can claim that it is indistinguishable from a complete reasoner like Pellet w.r.t. the LUBM TBox, regardless of the query and the data. In contrast, as shown in Table 1, all the other systems were found incomplete for some query.

We observe that all systems in Table 1 were incomplete for queries Q14-Q16, which involve undistinguished variables and hence require reasoning with existential quantifiers. Concerning the remaining queries, OWLim, Jena Micro in its OntModel implementation and Oracle OWLPrime, were found incomplete for $Q11(x) \leftarrow \mathsf{Student}(x)$. Even if $Q11$ contains only distinguished variables, LUBM's TBox entails the GCI $\mathsf{GradStudent} \sqsubseteq \mathsf{Student}$, which can only be derived using existential quantifiers. In contrast, DLEJena is complete for Q11 since it pre-computes the ontology's subsumption hierarchy using a complete DL reasoner before saturating the ABox. Finally, Sesame and Oracle RDFS, which are essentially RDF-Schema reasoners, are additionally incomplete for all queries that require reasoning with constructs not available in RDFS, such as inverse roles.

6 Conclusions

In this paper, we have studied the problem of query generation for evaluating the completeness of Semantic Web reasoners. Our techniques allow us to formally determine whether an incomplete system is indistinguishable from a complete one for a given TBox, regardless of the data and the query. Our framework and algorithms are complementary to those in [Stoilos *et al.*, 2010b] and [Stoilos *et al.*, 2010a], in which both TBox and query were assumed to be fixed.

Our evaluation is, however, still preliminary and we are planning to implement and evaluate our query generation techniques also for $\mathcal{EL}$. Finally, we are also planning to extend our techniques for more expressive DLs.

Acknowledgments

Research supported by the EU project SEALS (FP7-ICT-238975). Bernardo Cuenca Grau is also supported by a Royal Society University Research Fellowship.

[11] http://clarkparsia.com/pellet

References

[Baader *et al.*, 2002] F. Baader, D. McGuinness, D. Nardi, and P.F. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2002.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, 2005.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Automated Reasoning*, 39(3):385–429, 2007.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[Glimm *et al.*, 2007] Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proc. of IJCAI*, 2007.

[Grosof *et al.*, 2003] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of WWW*, pages 48–57, 2003.

[Guo *et al.*, 2005] Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. *J. Web Semantics (JWS)*, 3(2):158–182, 2005.

[Khalek and Khurshid, 2010] Shadi Abdul Khalek and Sarfraz Khurshid. Automated sql query generation for systematic testing of database engines. In *Proc. of ASE*, pages 329–332, 2010.

[Lutz *et al.*, 2009] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI*, 2009.

[Motik *et al.*, 2009] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz (Editors). OWL 2 Web Ontology Language Profiles. *W3C Recommendation*, 2009.

[Poess and Stephens, 2004] Meikel Poess and John M. Stephens. Generating thousand benchmark queries in seconds. In *Proc. of VLDB*, pages 1045–1053, 2004.

[Rosati, 2007] Riccardo Rosati. On conjunctive query answering in el. In *Proc. of DL*, 2007.

[Stoilos *et al.*, 2010a] Giorgos Stoilos, Bernardo Cuenca Grau, and Ian Horrocks. Completeness guarantees for incomplete reasoners. In *Proc. of ISWC*, 2010.

[Stoilos *et al.*, 2010b] Giorgos Stoilos, Bernardo Cuenca Grau, and Ian Horrocks. How incomplete is your semantic web reasoner? In *Proc. of AAAI*, pages 1431–1436, 2010.