Introducing Nominals to the Combined Query Answering Approaches for $\mathcal{EL}$

Giorgio Stefanoni and Boris Motik and Ian Horrocks
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road,
Oxford, OX1 3QD, UK

Abstract

So-called combined approaches answer a conjunctive query over a description logic ontology in three steps: first, they materialise certain consequences of the ontology and the data; second, they evaluate the query over the data; and third, they filter the result of the second phase to eliminate unsound answers. Such approaches were developed for various members of the DL-Lite and the $\mathcal{EL}$ families of languages, but none of them can handle ontologies containing nominals. In our work, we bridge this gap and present a combined query answering approach for $\mathcal{ELHO}_r^\perp$ — a logic that contains all features of the OWL 2 EL standard apart from transitive roles and complex role inclusions. This extension is nontrivial because nominals require equality reasoning, which introduces complexity into the first and the third step. Our empirical evaluation suggests that our technique is suitable for practical application, and so it provides a practical basis for conjunctive query answering in a large fragment of OWL 2 EL.

Introduction

Description logics (DLs) (Baader et al. 2007) are a family of knowledge representation formalisms that underpin OWL 2 (Cuenca Grau et al. 2008) — an ontology language used in advanced information systems with many practical applications. Answering conjunctive queries (CQs) over ontology-enriched data sets is a core reasoning service in such systems, so the computational aspects of this problem have received a lot of interest lately. For expressive DLs, the problem is at least doubly exponential in query size (Glimm et al. 2008). The problem, however, becomes easier for the $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) and the DL-Lite (Calvanese et al. 2007) families of DLs, which provide the foundation for the OWL 2 EL and the OWL 2 QL profiles of OWL 2. An important goal of this research was to devise not only worst-case optimal, but also practical algorithms. The known approaches can be broadly classified as follows.

The first group consists of automata-based approaches for DLs such as OWL 2 EL (Krötzsch, Rudolph, and Hitzler 2007) and Horn-$\mathcal{SHOIT}$ and Horn-$\mathcal{STROIT}$ (Ortiz, Rudolph, and Sînkus 2011). While worst-case optimal, these approaches are typically not suitable for practice since their best-case and worst-case performance often coincide.

The second group consists of rewriting-based approaches. Roughly speaking, these approaches rewrite the ontology and/or the query into another formalism, typically a union of conjunctive queries or a datalog program; the relevant answers can then be obtained by evaluating the rewriting over the data. Rewriting-based approaches were developed for members of the DL-Lite family (Calvanese et al. 2007; Artale et al. 2009), and the DLs $\mathcal{ELHIO}_r^\perp$ (Pérez-Urbina, Motik, and Horrocks 2010) and Horn-$\mathcal{SHIQ}$ (Eiter et al. 2012), to name just a few. A common problem, however, is that rewritings can be exponential in the ontology and/or query size. Although this is often not a problem in practice, such approaches are not worst-case optimal. An exception is the algorithm by Rosati (2007) that rewrites an $\mathcal{ELH}_r$ ontology into a datalog program of polynomial size; however, the algorithm also uses a nondeterministic step to transform the CQ into a tree-shaped one, and it is not clear how to implement this step in a goal-directed manner.

The third group consists of combined approaches, which use a three-step process: first, they augment the data with certain consequences of the ontology; second, they evaluate the CQ over the augmented data; and third, they filter the result of the second phase to eliminate unsound answers. The third step is necessary because, to ensure termination, the first step is unsound and may introduce facts that do not follow from the ontology; however, this is done in a way that makes the third step feasible. Such approaches have been developed for logics in the DL-Lite (Konchakov et al. 2011) and the $\mathcal{EL}$ (Lutz, Toman, and Wolter 2009) families, and they are appealing because they are worst-case optimal and practical: only the second step is intractable (in query size), but it can be solved using well-known database techniques.

None of the combined approaches proposed thus far, however, handles nominals — concepts containing precisely one individual. Nominals are included in OWL 2 EL, and they are often used to state that all instances of a class have a certain property value, such as ‘the sex of all men is male’, or ‘each German city is located in Germany’. In this paper we present a combined approach for $\mathcal{ELHO}_r^\perp$ — the DL that covers all features of OWL 2 EL apart from transitive roles and complex role inclusions. To the best of our knowledge, this is the first combined approach that handles nominals. Our extension is nontrivial because nominals require equality reasoning, which increases the complexity of the first and
the third step of the algorithm. In particular, nominals may introduce recursive dependencies in the filtering conditions used in the third phase; this is in contrast to the known combined approach for $\mathcal{EL}$ (Lutz, Toman, and Wolter 2009) in which filtering conditions are not recursive and can be incorporated into the input query. To solve this problem, our algorithm evaluates the original CQ and then uses a polynomial function to check the relevant conditions for each answer.

Following Krötzsch, Rudolph, and Hitzler (2008), instead of directly materialising the relevant consequences of the ontology and the data, we transform the ontology into a datalog program that captures the relevant consequences. Although seemingly just a stylistic issue, a datalog-based specification may be beneficial in practice: one can either materialise all consequences of the program bottom-up in advance, or one can use a top-down technique to compute only the consequences relevant for the query at hand. The latter can be particularly useful in information systems that have read-only access to the data, or where data changes frequently.

We have implemented a prototypical system using our algorithm, and we carried out a preliminary empirical evaluation of (i) the blowup in the number of facts introduced by the datalog program, and (ii) the number of answers obtained in the second phase. Our experiments show both of these numbers to be manageable in typical cases, suggesting that our algorithm provides a practical basis for answering CQs in an expressive fragment of OWL 2 EL.

The proofs of our technical results are provided in the technical report (Stefanović, Motik, and Horrocks 2013).

### Preliminaries

**Logic Programming.** We use the standard notions of variables, constants, function symbols, terms, atoms, formulas, and sentences (Fitting 1996). We often identify a conjunction with the set of its conjuncts. A substitution $\sigma$ is a partial mapping of variables to terms; $\text{dom}(\sigma)$ and $\text{rng}(\sigma)$ are the domain and the range of $\sigma$, respectively; $\sigma|_\alpha$ is the restriction of $\sigma$ to a set of variables $\alpha$; and, for $\alpha$ a term or a formula, $\sigma(\alpha)$ is the result of simultaneously replacing each free variable $x$ occurring in $\alpha$ with $\sigma(x)$. A Horn clause $C$ is an expression of the form $B_1 \land \ldots \land B_m \Rightarrow H$, where $H$ and each $B_i$ are atoms. Such a $C$ is a fact if $m = 0$, and it is commonly written as $H$. Furthermore, $C$ is safe if each variable occurring in $H$ also occurs in some $B_i$. A logic program $\Sigma$ is a finite set of safe Horn clauses; furthermore, $\Sigma$ is a datalog program if each clause in $\Sigma$ is function-free.

In this paper, we interpret a logic program $\Sigma$ in a model that can be constructed bottom-up. The Herbrand universe of $\Sigma$ is the set of all terms built from the constants and the function symbols occurring in $\Sigma$. Given an arbitrary set of facts $B$, let $\Sigma(B)$ be the smallest superset of $B$ such that, for each clause $\varphi \Rightarrow \psi \in \Sigma$ and each substitution $\sigma$ mapping the variables occurring in the clause to the Herbrand universe of $\Sigma$, if $\varphi(\sigma) \in B$, then $\psi(\sigma) \in \Sigma(B)$. Let $I_0$ be the set of all facts occurring in $\Sigma$; for each $i \in \mathbb{N}$, let $I_{i+1} = \Sigma(I_i)$; and let $I = \bigcup_{i \in \mathbb{N}} I_i$. Then $I$ is the minimal Herbrand model of $\Sigma$, and it is well known that $I$ satisfies $\forall \vec{x}. C$ for each Horn clause $C \in \Sigma$ and $\vec{x}$ the vector of all variables occurring in $C$.

<table>
<thead>
<tr>
<th>Type</th>
<th>Axiom</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${a} \subseteq A$</td>
<td>$\sim \rightarrow A(a)$</td>
</tr>
<tr>
<td>2</td>
<td>$A \subseteq B$</td>
<td>$\sim \rightarrow A(x) \rightarrow B(x)$</td>
</tr>
<tr>
<td>3</td>
<td>$A \subseteq {a}$</td>
<td>$\sim \rightarrow A(x) \rightarrow x \approx a$</td>
</tr>
<tr>
<td>4</td>
<td>$A \land A_2 \subseteq A$</td>
<td>$\sim \rightarrow A_1(x) \land A_2(x) \rightarrow A(x)$</td>
</tr>
<tr>
<td>5</td>
<td>$\exists R. A_1 \subseteq A$</td>
<td>$\sim \rightarrow R(x,y) \land A_1(y) \rightarrow A(x)$</td>
</tr>
<tr>
<td>6</td>
<td>$A_1 \subseteq \exists R.A$</td>
<td>$\sim \rightarrow A_1(x) \rightarrow \exists R(x,f_{R,A}(x)) \rightarrow A_1(x) \rightarrow A(f_{R,A}(x))$</td>
</tr>
<tr>
<td>7</td>
<td>$R \subseteq S$</td>
<td>$\sim \rightarrow R(x,y) \rightarrow S(x,y)$</td>
</tr>
<tr>
<td>8</td>
<td>range$(R,A)$</td>
<td>$\sim \rightarrow (x,y) \rightarrow A(y)$</td>
</tr>
</tbody>
</table>

*Table 1: Transforming $\mathcal{ELHO}_\pi$ Axioms into Horn Clauses*

In this paper we allow a logic program $\Sigma$ to contain the equality predicate $\approx$. In first-order logic, $\approx$ is usually interpreted as the identity over the interpretation domain; however, $\approx$ can also be explicitly axiomatised (Fitting 1996). Let $\Sigma_\approx$ be the set containing clauses (1)–(3), an instance of clause (4) for each $n$-ary predicate $R$ occurring in $\Sigma$ and each $1 \leq i \leq n$, and an instance of clause (5) for each $n$-ary function symbol $f$ occurring in $\Sigma$ and each $1 \leq i \leq n$.

$\rightarrow x \approx x$ \hspace{1cm} (1)

$\rightarrow x \approx x \rightarrow x \approx x$ \hspace{1cm} (2)

$\rightarrow x_1 \approx x_2 \rightarrow x_2 \approx x_3 \rightarrow x_1 \approx x_3$ \hspace{1cm} (3)

$\rightarrow R(\vec{x}) \land \rightarrow x_1 \approx x_1' \rightarrow R(x_1, x_1', x_2, x_2', \ldots, x_n)$ \hspace{1cm} (4)

$\rightarrow x_1 \approx x_1' \rightarrow f(\ldots, x_1, \ldots) \approx f(\ldots, x_1', \ldots)$ \hspace{1cm} (5)

The minimal Herbrand model of a logic program $\Sigma$ that contains $\approx$ is the minimal Herbrand model of $\Sigma \cup \Sigma_\approx$.

**Conjunctive Queries.** A conjunctive query (CQ) is a formula $q = \exists \vec{y}. \psi(\vec{x}, \vec{y})$ with $\psi$ a conjunction of function-free atoms over variables $\vec{x} \cup \vec{y}$. Variables $\vec{x}$ are the answer variables of $q$. Let $N_q(q)$ be the set of terms occurring in $q$.

Let $\tau$ be a substitution such that $\text{rng}(\tau)$ contains only constants. Then, $\tau(q) = \exists \vec{z}. \tau(\psi)$, where $\vec{z}$ is obtained from $\vec{y}$ by removing each variable $y \in \vec{y}$ for which $\tau(y)$ is defined. Note that, according to this definition, non-free variables can also be replaced; for example, given $q = \exists y_1, y_2. R(y_1, y_2)$ and $\tau = \{y_2 \rightarrow a\}$, we have $\tau(q) = \exists y_1. R(y_1, a)$.

Let $\Sigma$ be a logic program, let $I$ be the minimal Herbrand model of $\Sigma$, and let $q = \exists \vec{y}. \psi(\vec{x}, \vec{y})$ be a CQ that uses only the predicates occurring in $\Sigma$. A substitution $\pi$ is a candidate answer for $q$ in $\Sigma$ if $\text{dom}(\pi) = \vec{x}$ and $\text{rng}(\pi)$ contains only constants; furthermore, such a $\pi$ is a certain answer to $q$ over $\Sigma$, written $\Sigma \models \pi(q)$, if a substitution $\tau$ exists such that $\text{dom}(\tau) = \vec{x} \cup \vec{y}$, $\pi = \tau|_{\vec{x}}$, and $\tau(q) \subseteq I$.

**Description Logic.** DL $\mathcal{ELHO}_\pi^*$ is defined w.r.t. a signature consisting of mutually disjoint and countably infinite sets $N_C$, $N_R$, and $N_I$ of atomic concepts (i.e., unary predicates), roles (i.e., binary predicates), and individuals (i.e., constants), respectively. Furthermore, for each individual $a \in N_I$, expression $\{a\}$ denotes a nominal—that is, a concept containing precisely the individual $a$. Also, we assume that $\top$ and $\bot$ are unary predicates (without any predefined meaning) not occurring in $N_C$. We consider only normalised knowledge bases, as it is well known (Baader, Brandt, and Lutz 2005) that each $\mathcal{ELHO}_\pi^*$ knowledge base can be normalised in polynomial time without affecting the answers to CQs. An $\mathcal{ELHO}_\pi^*$ TBox is a finite set of ax-
ions of the form shown in the left-hand side of Table 1, where \( A_{\text{a}} \in N_C \cup \{ \top \} \), \( B \in N_C \cup \{ \bot \} \), \( R,S \in N_R \), and \( a \in N_I \). An ABox \( A \) is a finite set of facts constructed using the symbols from \( N_C \cup \{ \top, \bot \}, N_R, \) and \( N_I \). Finally, an \( \text{ELCHO}^\circ \) knowledge base (KB) is a tuple \( \mathcal{K} = (T,A) \), where \( T \) is an \( \text{ELCHO}^\circ \) TBox and \( A \) is an ABox such that each predicate occurring in \( A \) also occurs in \( T \).

We interpret \( T \) as a logic program. Table 1 shows how to translate a TBox \( T \) into a logic program \( \Xi(T) \). Moreover, let \( \top(T) \) be the set of the following clauses instantiated for each atomic concept \( A \) and each role \( R \) occurring in \( T \).

\[
A(x) \rightarrow \top(x) \quad R(x,y) \rightarrow \top(x) \quad R(x,y) \rightarrow \top(y)
\]

A knowledge base \( \mathcal{K} = (T,A) \) is translated into the logic program \( \Xi(\mathcal{K}) = \Xi(T) \cup \top(T) \cup A \). Then, \( \mathcal{K} \) is unsatisfiable if \( \Xi(\mathcal{K}) \models \exists y. \bot(y) \). Furthermore, given a conjunctive query \( q \) and a candidate answer \( \pi \) for \( q \), we write \( \mathcal{K} \models \pi(q) \) iff \( \mathcal{K} \) is unsatisfiable or \( \Xi(\mathcal{K}) \models \pi(q) \).


Datalog Rewriting of \( \text{ELCHO}^\circ \) TBoxes

For the rest of this section, we fix an arbitrary \( \text{ELCHO}^\circ \) knowledge base \( \mathcal{K} = (T,A) \). We next show how to transform \( T \) into a datalog program \( D(\mathcal{K}) \) that can be used to check the satisfiability of \( \mathcal{K} \). In the following section, we then show how to use \( D(\mathcal{K}) \) to answer conjunctive queries.

Due to axioms of type 6 (cf. Table 1), \( \Xi(\mathcal{K}) \) may contain function symbols and is generally not a datalog program; thus, the evaluation of \( \Xi(\mathcal{K}) \) may not terminate. To ensure termination, we eliminate function symbols from \( \Xi(\mathcal{K}) \) using the technique by Krötzsch, Rudolph, and Hitzler (2008); for each \( A \in N_C \cup \{ \top \} \) and each \( R \in N_R \) occurring in \( T \), we introduce a globally fresh and unique auxiliary individual \( o_{R,A} \). Intuitively, \( o_{R,A} \) represents all terms in the Herbrand universe of \( \Xi(\mathcal{K}) \) needed to satisfy the existential concept \( \exists R.A \). Krötzsch, Rudolph, and Hitzler (2008) used this technique to facilitate taxonomic reasoning, while we use it to obtain a practical CQ answering algorithm. Please note that \( o_{R,A} \) depends on both \( R \) and \( A \), whereas in the known approaches such individuals depend only on \( A \) (Lutz, Toman, and Wolter 2009) or \( R \) (Kontchakov et al. 2011).

**Definition 1.** Datalog program \( D(T) \) is obtained by translating each axiom of type other than 6 in the TBox \( T \) of \( \mathcal{K} \) into a clause as shown in Table 1, and by translating each axiom \( A_1 \subseteq 3R.A \) in \( T \) into clauses \( A_1(x) \rightarrow R(x,o_{R,A}) \) and \( A_1(x) \rightarrow A(o_{R,A}) \). Furthermore, the translation of the \( \text{翎k} \) into datalog is given by \( D(\mathcal{K}) = D(T) \cup \top(T) \cup A \).

**Example 1.** Let \( T \) be the following \( \text{ELCHO}^\circ \) TBox:

- \( \text{KRC} \subseteq \text{Taught.JProf} \quad \text{Taught.T} \subseteq \text{Course} \)
- \( \text{Course} \subseteq \text{Taught.Prof} \quad \{ \text{kr} \} \subseteq \text{KRC} \)
- \( \text{Prof} \subseteq \text{Advisor.Prof} \quad \text{KRC} \subseteq \text{Course} \)
- \( \text{JProf} \subseteq \{ \text{john} \} \quad \text{range}(\text{Taught}, \text{Prof}) \)

Figure 1: Representing the Models of \( \Xi(\mathcal{K}) \).

Then, \( D(T) \) contains the following clauses:

- \( \text{KRC}(x) \rightarrow \text{taught}(x, o_{T,J}) \)
- \( \text{JProf}(x) \rightarrow x \approx \text{john} \)
- \( \text{KRC}(x) \rightarrow \text{JProf}(o_{T,J}) \quad \text{taught}(x,y) \rightarrow \text{Course}(x) \)
- \( \text{Course}(x) \rightarrow \text{taught}(x, o_{T,P}) \quad \text{KRC}(kr) \)
- \( \text{Course}(x) \rightarrow \text{Prof}(o_{T,P}) \quad \text{KRC}(x) \rightarrow \text{Course}(x) \)
- \( \text{Prof}(x) \rightarrow \text{Advisor}(x, o_{A,P}) \quad \text{taught}(x,y) \rightarrow \text{Prof}(y) \)
- \( \text{Prof}(x) \rightarrow \text{Prof}(o_{A,P}) \)

The following result straightforwardly follows from the definition of \( \Xi(\mathcal{K}) \) and \( D(\mathcal{K}) \).

**Proposition 2.** Program \( D(\mathcal{K}) \) can be computed in time linear in the size of \( \mathcal{K} \).

Next, we prove that the datalog program \( D(\mathcal{K}) \) can be used to decide the satisfiability of \( \mathcal{K} \). To this end, we define a function \( \delta \) that maps each term \( w \) in the Herbrand universe of \( \Xi(\mathcal{K}) \) to the Herbrand universe of \( D(\mathcal{K}) \) as follows:

\[
\delta(w) = \begin{cases} 
  w & \text{if } w \in N_I, \\
  o_{R,A} & \text{if } w \text{ is of the form } w = f_{R,A}(w').
\end{cases}
\]

Let \( I \) and \( J \) be the minimal Herbrand models of \( \Xi(\mathcal{K}) \) and \( D(\mathcal{K}) \), respectively. Mapping \( \delta \) establishes a tight relationship between \( I \) and \( J \) as illustrated in the following example.

**Example 2.** Let \( A = \{ \text{Course}(ai) \} \), let \( T \) be as in Example 1, and let \( \mathcal{K} = (T,A) \). Figure 1 shows a graphical representation of the minimal Herbrand models \( I \) and \( J \) of \( \Xi(\mathcal{K}) \) and \( D(\mathcal{K}) \), respectively. The grey dotted lines show how \( \delta \) relates the terms in \( I \) to the terms in \( J \). For the sake of clarity, Figure 1 does not show the reflexivity of \( \approx \).

Mapping \( \delta \) is a homomorphism from \( I \) to \( J \).

**Lemma 3.** Let \( I \) and \( J \) be the minimal Herbrand models of \( \Xi(\mathcal{K}) \) and \( D(\mathcal{K}) \), respectively. Mapping \( \delta \) satisfies the following three properties for all terms \( w' \) and \( w \), each \( B \in N_C \cup \{ \top, \bot \} \), and each \( R \in N_R \):

1. \( B(w) \in I \) implies \( B(\delta(w)) \in J \).
2. \( R(w', w) \in I \) implies \( R(\delta(w'), \delta(w)) \in J \).
3. \( w' \approx w \in I \) implies \( \delta(w') \approx \delta(w) \in J \).

For a similar result in the other direction, we need a couple of definitions. Let \( H \) be an arbitrary Herbrand model. Then,
dom(H) is the set containing each term w that occurs in H in at least one fact with a predicate in N_C ∪ {⊤, ⊥} ∪ N_R; note that, by this definition, we have w /∈ dom(H) whenever w occurs in H only in assertions involving the ≈ predicate. Furthermore, aux_H is the set of all terms w ∈ dom(H) such that, for each term w' with w ≈ w' ∈ H, we have w' /∈ N_I. We say that the terms in aux_H are ‘true’ auxiliary terms— that is, they are not equal to an individual in N_I. In Figure 1, bold terms are ‘true’ auxiliary terms in I and J.

**Lemma 4.** Let I and J be the minimal Herbrand models of Ξ(K) and D(K), respectively. Mapping δ satisfies the following five properties for all terms w_1 and w_2 in dom(I), each B ∈ N_C ∪ {⊤, ⊥}, and each R ∈ N_R.

1. B(δ(w_1)) ∈ J implies that B(w_1) ∈ I.
2. R(δ(w_1), δ(w_2)) ∈ J and δ(w_2) /∈ aux_J imply that
   R(w_1, w_2) ∈ I.
3. R(δ(w_1), δ(w_2)) ∈ J and δ(w_2) ∈ aux_J imply that
   δ(w_2) is of the form o_{P,A}(w_1) ∈ I, and that a term w'_1 exists such that R(w'_1, w_2) ∈ I.
4. δ(w_1) ≈ w_2 ∈ J and δ(w_2) /∈ aux_J imply that
   w_1 ≈ w_2 ∈ I.
5. For each term u occurring in J, term w ∈ dom(I) exists such that δ(u) = w.

Lemmas 3 and 4 allow us to decide the satisfiability of K by answering a simple query over D(K), as shown in Proposition 5. The complexity claim is due to the fact that each clause in D(K) contains a bounded number of variables (Dantsin et al. 2001).

**Proposition 5.** For K, an arbitrary ELC_H knowledge base, Ξ(K) = ⊢ y⊥(y) if and only if D(K) = ⊢ y⊥(y).

Furthermore, the satisfiability of K can be checked in time polynomial in the size of K.

**Answering Conjunctive Queries**

In this section, we fix a satisfiable ELC_H knowledge base K = (T, A) and a conjunctive query q = ⊢ y⊥(y). Furthermore, we fix I and J to be the minimal Herbrand models of Ξ(K) and D(K), respectively.

While D(K) can be used to decide the satisfiability of K, the following example shows that D(K) cannot be used directly to compute the answers to q.

**Example 3.** Let K be as in Example 2, and let q_1, q_2, and q_3 be the following conjunctive queries:

q_1 = “taught(x_1, x_2)”

q_2 = “∃y_1, y_2, y_3. taught(x_1, y_1) ∧ taught(x_2, y_2) ∧ advisor(y_1, y_3) ∧ advisor(y_2, y_3)”

q_3 = “∃y. advisor(y, y)”

Furthermore, let τ_i be the following substitutions:

τ_1 = {x_1 → kr, x_2 → o_T,P}

τ_2 = {x_1 → kr, x_2 → a_i, y_1 → o_T,P, y_2 → o_T,P, y_3 → o_A,P}

τ_3 = {y → o_A,P}

Finally, let each π_i be the projection of τ_i to the answer variables of q_i. Using Figure 1, one can readily check that D(K) ⊨ τ_i(q_i), but Ξ(K) ⊭ τ_i(q_i), for each 1 ≤ i ≤ 3. ⊓⊔

This can be explained by observing that J is a homomorphic image of I. Now homomorphisms preserve CQ answers (i.e., Ξ(K) ⊨ τ(q) implies D(K) ⊨ τ(q)), but they can also introduce unsound answers (i.e., D(K) ⊨ τ(q) does not necessarily imply Ξ(K) ⊨ τ(q)). This gives rise to the following notion of spurious answers.

**Definition 6.** A substitution τ with dom(τ) = x̃ ∪ ỹ and D(K) ⊨ τ(q) is a spurious answer to q if τ|_x̃ is not a certain answer to q over Ξ(K).

Based on these observations, we answer q over K in two steps: first, we evaluate q over D(K) and thus obtain an over-estimation of the certain answers to q over Ξ(K); second, for each substitution τ obtained in the first step, we eliminate spurious answers using a special function iSpur. We next formally introduce this function. We first present all relevant definitions, after which we discuss the intuitions. As we shall see, each query in Example 3 illustrates a distinct source of spuriousness that our function needs to deal with.

**Definition 7.** Let τ be a substitution s.t. dom(τ) = x̃ ∪ ỹ and D(K) ⊨ τ(q). Relation ⊥ ⊂ N_T(q) × N_T(q) for q, r, and D(K) is the smallest reflexive, symmetric, and transitive relation closed under the fork rule, where aux_D(K) is the set containing each individual u from D(K) for which no individual c ∈ N_I exists such that D(K) |= u ≈ c.

(fork) s′ ⋲ t′ R(s, s′) and P(t, t′) in q, and τ(s′) ∈ aux_D(K).

Please note that the definition aux_D(K) is actually a reformulation of the definition of aux_J, but based on the consequences of D(K) rather than the facts in J.

Relation ⊥ is reflexive, symmetric, and transitive, so it is an equivalence relation, which allows us to normalise each term t ∈ N_T(q) to a representative of its equivalence class using the mapping γ defined below. We then construct a graph G_aux that checks whether substitution τ matches ‘true’ auxiliary individuals in a way that cannot be converted to a match over ‘true’ auxiliary terms in I.

**Definition 8.** Let τ and ⊥ be as specified in Definition 7. Function γ : N_T(q) → N_T(q) maps each term t ∈ N_T(q) to an arbitrary, but fixed representative γ(t) of the equivalence class of ⊥ that contains t. Furthermore, the directed graph G_aux = (V_aux, E_aux) is defined as follows.

- Set V_aux contains a vertex γ(t) ∈ N_T(q) for each term t ∈ N_T(q) such that τ(t) ∈ aux_D(K).
- Set E_aux contains an edge (γ(s), γ(t)) for each atom of the form R(s, t) in q such that {γ(s), γ(t)} ⊂ V_aux.

Query q is aux-cyclic w.r.t. τ and D(K) if G_aux contains a cycle; otherwise, q is aux-acyclic w.r.t. τ and D(K).

We are now ready to define our function that checks whether a substitution τ is a spurious answer.

**Definition 9.** Let τ and ⊥ be as specified in Definition 7. Then, function iSpur(q, D(K), τ) returns t if and only if at least one of the following conditions hold.

(a) Variable x ∈ x̃ exists such that τ(x) /∈ N_I.

(b) Terms s and t occurring in q exist such that s ⋲ t and D(K) |= τ(s) ⋲ τ(t).
(c) Query $q$ is aux-cyclic w.r.t. $\tau$ and $D(\mathcal{K})$.

We next discuss the intuition behind our definitions. We ground our discussion in minimal Herbrand models $I$ and $J$, but our technique does not depend on such models: all conditions are stated as entailments that can be checked using an arbitrary sound and complete technique. Since $\mathcal{K}$ is an $\mathcal{ELHO}^*_2$ knowledge base, model $I$ is forest-shaped: roughly speaking, the role assertions in $I$ that involve at least one functional term are of the form $R(w_1, f_{R,A}(w_1))$ or $R(w_1, a)$ for $a \in N_I$; thus, $I$ can be viewed as a family of directed trees whose roots are the individuals in $N_I$ and whose edges point from parents to children or to the individuals in $N_I$. This is illustrated in Figure 1, whose lower part shows the the forest-model of the knowledge base from Example 3. Note that assertions of the form $R(w_1, a)$ are mapped to ‘true’ auxiliary individuals (mapping $\gamma$ simply ensures that equal terms are represented as one vertex).

Now let $\tau$ be a substitution such that $D(\mathcal{K}) \models \tau(q)$, and let $\pi = \tau|_F$. If $\tau$ is not a spurious answer, it should be possible to convert $\tau$ into a substitution $\pi^*$ such that $\pi = \pi^*|_x$ and $\pi^*(q) \subseteq I$. Using the queries from Example 3, we next identify three reasons why this may not be possible.

First, $\tau$ may map an answer variable of $q$ to an auxiliary individual, so by the definition $\pi$ cannot be a certain answer to $q$; condition (a) of Definition 9 identifies such cases. Query $q_1$ and substitution $\tau_1$ from Example 3 illustrate such a situation: $\tau_2(x_2) = o_{T,P}$ and $o_{T,P}$ is a ‘true’ auxiliary individual, so $\pi_1$ is not a certain answer to $q_1$.

The remaining two problems arise because model $J$ is not forest-shaped, so $\pi$ might map $q$ into $I$ in a way that cannot be converted into a substitution $\pi^*$ that maps $q$ into $I$.

The second problem is best explained using substitution $\tau_2$ and query $q_2$ from Example 3. Query $q_2$ contains a ‘fork’ advisor$(y_1, y_3)$ $\land$ advisor$(y_2, y_3)$. Now $\tau_2(y_3) = o_{A,P}$ is a ‘true’ auxiliary individual, and so it represents ‘true’ auxiliary terms $f_{A,P}(f_{T,P}(ai))$, $f_{A,P}(f_{T,P}(kr))$, and so on. Since $J$ is forest-shaped, a match $\pi_2^*$ for $q$ in $J$ obtained from $\tau_2$ would need to map $y_3$ to one of these terms; let us assume that $\pi_2^*(y_3) = f_{A,P}(f_{T,P}(ai))$. Since $J$ is forest-shaped and $f_{A,P}(f_{T,P}(ai))$ is a ‘true’ auxiliary term, this means that both $y_1$ and $y_2$ must be mapped to the same term (in both $J$ and $I$). This is captured by the (fork) rule: in our example, the rule derives $y_1 \sim y_2$, and condition (b) of Definition 9 checks whether $\tau_2$ maps $y_1$ and $y_2$ in a way that satisfies this constraint. Note that, due to role hierarchies, the rule needs to be applied to atoms $R(s, s')$ and $P(t, t')$ with $R \neq P$. Moreover, such constraints must be propagated further up the query. In our example, due to $y_1 \sim y_2$, atoms $taught(x_1, y_1) \land taught(x_2, y_2)$ in $q_2$ also constitute a ‘fork’, so the rule derives $x_1 \sim x_2$; now this allows condition (b) of Definition 9 to correctly identify $\tau_2$ as spurious.

The third problem is best explained using substitution $\tau_3$ and query $q_3$ from Example 3. Model $J$ contains a ‘loop’ on individual $o_{A,P}$, which allows $\tau_3$ to map $q_3$ into $J$. In contrast, model $I$ is forest-shaped, and so the ‘true’ auxiliary terms that correspond to $o_{A,P}$ do not form loops. Condition (c) of Definition 9 detects such situations using the graph $G_{aux}$. The vertices of $G_{aux}$ correspond to the terms of $q$ that are matched to ‘true’ auxiliary individuals (mapping $\gamma$ simply ensures that equal terms are represented as one vertex), and edges of $G_{aux}$ correspond to the role atoms in $q$. Hence, if $G_{aux}$ is cyclic, then the substitution $\pi^*$ obtained from $\tau$ would need to match the query $q$ over a cycle of ‘true’ auxiliary terms, which is impossible since $I$ is forest-shaped.

Unlike the known combined approaches, our approach does not extend $q$ with conditions that detect spurious answers. Due to nominals, the relevant equality constraints have a recursive nature, and they depend on both the substitution $\tau$ and on the previously derived constraints. Consequently, filtering in our approach is realised as postprocessing; furthermore, to ensure correctness of our filtering condition, auxiliary individuals must depend on both a role and an atomic concept. The following theorem proves the correctness of our approach.

**Theorem 10.** Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a satisfiable $\mathcal{ELHO}^*_2$ KB, let $q = \exists y . \psi(x, y)$ be a $CQ$, and let $\tau : x \mapsto N_I$ be a candidate answer for $q$. Then, $\exists(\mathcal{K}) \models \pi(q)$ iff a substitution $\tau$ exists such that $\text{dom}(\tau) = \bar{x} \cup \bar{y}$, $\tau|_x = \pi$, $D(\mathcal{K}) \models \tau(q)$, and $\text{isSpur}(q, D(\mathcal{K}), \tau) = 0$.

Furthermore, isSpur$(q, D(\mathcal{K}), \tau)$ can be evaluated in polynomial time, so the main source of complexity in our approach is in deciding whether $D(\mathcal{K}) \models \tau(q)$ holds. This gives rise to the following result.

**Theorem 11.** Deciding whether $\mathcal{K} \models \tau(q)$ holds can be implemented in nondeterministic polynomial time w.r.t. the size of $\mathcal{K}$ and $q$, and in polynomial time w.r.t. the size of $\mathcal{A}$.

**Evaluation**

To gain insight into the practical applicability of our approach, we implemented our technique in a prototypical system. The system uses HermiT, a widely used ontology reasoner, as a datalog engine in order to materialise the consequences of $D(\mathcal{K})$ and evaluate $q$. The system has been implemented in Java, and we ran our experiments on a MacBook Pro with 4GB of RAM and an Intel Core 2 Duo 2.4 Ghz processor. We used two ontologies in our evaluation, details of which are given below. The ontologies, queries, and the prototype system are all available online at http://www.cs.qc.edu/igis/tools/KARMA/.

The LSTW benchmark (Lutz et al. 2012) consists of an OWL 2 QL version of the LUBM ontology (Guo, Pan, and Heflin 2005), queries $q_1, \ldots, q_{11}$, and a data generator. The LSTW ontology extends the standard LUBM ontology with several axioms of type 6 (see Table 1). To obtain an $\mathcal{ELHO}^*_2$ ontology, we removed inverse roles and datatypes, added 11 axioms using 9 freshly introduced nominals, and added one

| L-5 | Mat. | 100848 | 100868 (0.01) | 169079 | 309350 (0.01) | 296941 | 632489 (49.2) |
| L-10 | Mat. | 202487 | 202407 (0.01) | 339746 | 621158 (0.01) | 598695 | 1277575 (49.3) |
| L-20 | Mat. | 426144 | 420164 (0.01) | 714692 | 1304815 (0.01) | 1259936 | 2691766 (49.3) |
| SEM | Mat. | 17945 | 17953 (0.04) | 17945 | 25608 (0.03) | 47248 | 76590 (38.3) |

Table 2: Size of the materialisations.
axiom of type 4 (see Table 1). These additional axioms resemble the ones in Example 1, and they were designed to test equality reasoning. The resulting signature consists of 132 concepts, 32 roles, and 9 nominals, and the ontology contains 180 axioms. From the 11 LSTW queries, we did not consider queries $q_1^4$, $q_6^9$, $q_2^5$, and $q_{14}^1$ because their result sets were empty: $q_5^1$ relies on existential quantification over inverse roles, and the other three are empty already w.r.t. the original LSTW ontology. Query $q_5^2$ is similar to query $q_2$ from Example 3, and it was designed to produce only spurious answers and thus stress the system. We generated data sets with 5, 10 and 20 universities. For each data set, we denote with L-i the knowledge base consisting of our $\mathcal{ELH}O^E_i$ ontology and the ABox for i universities (see Table 2).

SEMINTEC is an ontology about financial services developed within the SEMINTEC project at the University of Poznan. To obtain an $\mathcal{ELH}O^E_i$ ontology, we removed inverse roles, role functionality axioms, and universal restrictions, added nine axioms of type 6 (see Table 1), and added six axioms using 4 freshly introduced nominals. The resulting ontology signature consists of 60 concepts, 16 roles, and 4 nominals, and the ontology contains 173 axioms. Queries $q_5^3$-$q_5^4$ are tree-shaped queries used in the SEMINTEC project, and we developed queries $q_6^9$-$q_6^{10}$ ourselves. Query $q_5^6$ resembles query $q_2$ from LSTW, and queries $q_6^8$ and $q_6^9$ were designed to retrieve a large number of answers containing auxiliary individuals, thus stressing condition (a) of Definition 9. Finally, the SEMINTEC ontology comes with a data set consisting of approximately 65,000 facts concerning 18,000 individuals (see row SEM in Table 2).

The practicality of our approach, we believe, is determined mainly by the following two factors. First, the number of facts involving auxiliary individuals introduced during the materialisation phase should not be ‘too large’. Table 2 shows the materialisation results: the first column shows the number of individuals before and after materialisation and the percentage of ‘true’ auxiliary individuals, the second column shows the number of unary facts before and after materialisation and the percentage of facts involving a ‘true’ auxiliary individual, and the third column does the same for binary facts. As one can see, for each input data set, the materialisation step introduces few ‘true’ auxiliary individuals, and the number of facts at most doubles. The number of unary facts involving a ‘true’ auxiliary individual does not change with the size of the input data set, whereas the number of such binary facts increases by a constant factor. This is because, in clauses of type 6, atoms $A(o_R,A)$ do not contain a variable, whereas atoms $R(x,o_R,A)$ do.

Second, evaluating $q$ over $D(K)$ should not produce too many spurious answers. Table 3 shows the total number of answers for each query—that is, the number of answers obtained by evaluating the query over $D(K)$; furthermore, the table also shows what percentage of these answers are spurious. Queries $q_6^5$, $q_6^{10}$, $q_6^9$, and $q_6^5$ retrieve a significant percentage of spurious answers. However, only query $q_6^5$ has proven to be challenging for our system due to the large number of retrieved answers, with an evaluation time of about 40 minutes over the largest knowledge base (L-20). Surprisingly, $q_6^4$ also performed rather poorly despite a low number of spurious answers, with an evaluation time of about 20 minutes for L-20. All other queries were evaluated in at most a few seconds, thus suggesting that queries $q_6^4$ and $q_6^5$ are problematic mainly because Hermit does not implement query optimisation algorithms typically used in relational databases.

### Conclusion

We presented the first combined technique for answering conjunctive queries over DL ontologies that include nominals. A preliminary evaluation suggests the following. First, the number of materialised facts over ‘true’ anonymous individuals increases by a constant factor with the size of the data. Second, query evaluation results have shown that, while some cases may be challenging, in most cases the percentage of answers that are spurious is manageable. Hence, our technique provides a practical CQ answering algorithm for a large fragment of OWL 2 EL.

We anticipate several directions for our future work. First, we would like to investigate the use of top-down query evaluation techniques, such as magic sets (Abiteboul, Hull, and Vianu 1995) or SLG resolution (Chen and Warren 1993). Second, tighter integration of the detection of spurious answers with the query evaluation algorithms should make it possible to eagerly detect spurious answers (i.e., before the query is fully evaluated). Lutz et al. (2012) already implemented a filtering condition as a user-defined function in a database, but it is unclear to what extent such an implementation can be used to optimise query evaluation. Finally, we would like to extend our approach to all of OWL 2 EL.

### Acknowledgements

This work was supported by the Royal Society; Alcatel-Lucent; the EU FP7 project OPTIQUE; and the EPSRC projects ExODA, MASI$^3$, and QueRe.

### References


Stefanoni, G.; Motik, B.; and Horrocks, I. 2013. Introducing Nominals to the Combined Query Answering Approaches for EL. *CoRR* abs/1303.7430.