# A Practical Approach for Scalable Conjunctive Query Answering on Acyclic $\mathcal{EL}^+$ Knowledge Base

Jing Mei[1], Shengping Liu[1], Guotong Xie[1], Aditya Kalyanpur[2], Achille Fokoue[2], Yuan Ni[1], Hanyu Li[1], and Yue Pan[1]

[1] IBM China Research Lab, Building 19 ZGC Software Park, Beijing 100193, China
{meijing,liusp,xieguot,niyuan,lihanyu,panyue}@cn.ibm.com
[2] IBM Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA
{adityakal,achille}@us.ibm.com

**Abstract.** Conjunctive query answering for $\mathcal{EL}^{++}$ ontologies has recently drawn much attention, as the Description Logic $\mathcal{EL}^{++}$ captures the expressivity of many large ontologies in the biomedical domain and is the foundation for the OWL 2 EL profile. In this paper, we propose a practical approach for conjunctive query answering in a fragment of $\mathcal{EL}^{++}$, namely acyclic $\mathcal{EL}^+$, that supports role inclusions. This approach can be implemented with low cost by leveraging any existing relational database management system to do the ABox data completion and query answering. We conducted a preliminary experiment to evaluate our approach using a large clinical data set and show our approach is practical.

## 1 Introduction

The OWL 2 EL profile is a subset of OWL 2 that captures the expressive power of many large ontologies in the biomedical domain, such as the Gene Ontology [6], the Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT) [9], and large parts of the Galen Medical Knowledge Base ontology [13]. This profile is based on the Description Logic language $\mathcal{EL}^{++}$ [1,2] for which the concept subsumption and instance checking problem can be decided in polynomial time.

Meanwhile, conjunctive query answering, which originated from research in relational databases (RDB), is becoming a crucial requirement for ontology-based, data intensive applications. In biomedicine, there are several, very large $\mathcal{EL}^+$ ontological datasets, e.g., protein data annotated with the Gene Ontology and clinical data annotated using SNOMED CT. However, theoretical results have proved that conjunctive query answering is undecidable in both $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$ [14]. To regain the decidability for addressing conjunctive queries, some fragments of $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$ need to be identified. One approach is to exclude role inclusions, e.g., related work on $\mathcal{EL}$ [11] and its follow-up work on $\mathcal{ELH}^{dr}_{\perp}$ [12], while another approach is to impose restrictions on role inclusions, e.g., the so-called regular $\mathcal{EL}^{++}$ [10]. In either of the two approaches, less attention is paid to provide full support for role inclusions. Considering that role inclusion does play an important role in biomedical ontologies (such as transitivity for the *part-of* relationship in the Gene Ontology and the right identity axiom for *direct-substance* $\circ$ *active-ingredient* $\sqsubseteq$ *direct-substance* in SNOMED CT), we prefer to keep expressive roles in the logic and explore an alternate solution.

In this paper, we present a practical approach for scalable conjunctive query answering on acyclic $\mathcal{EL}^+$ KBs. The *acyclic* notion generalizes the classical one (as defined in the DL handbook [4]), by including general concept inclusions. Note that acyclic $\mathcal{EL}^+$ KBs can admit the irregularity in $\mathcal{EL}^+$ role inclusions. Also, acyclic $\mathcal{EL}^+$ is really useful in practice, because it is expressive enough for the commonly used biomedical ontologies, e.g., Gene Ontology, SNOMED CT.

At a glance, our approach is similar to the bottom-up rule evaluation approach which consists of: (1) doing an ABox completion to precompute all inferred assertions; (2) answering conjunctive queries on the completed ABox. In the first phase, we adopt the idea of "shared" individuals for existential restrictions, referred to as *canonical individuals* in this paper. For example, suppose $A \sqsubseteq \exists R.B$ in the TBox with $A(a)$ and $A(b)$ in the ABox, we will generate one and only one canonical individual $u$ w.r.t. the existential restriction $\exists R.B$, and the completed ABox will have the assertions $B(u)$, $R(a, u)$, and $R(b, u)$. Thus, the cost of inference and the size of the completed dataset is greatly reduced, due to less fresh individuals. However, the canonical individual $u$ is now "shared" by both $a$ and $b$. If a query asks for $R(x_1, y)$ and $R(x_2, y)$, then the bindings $x_1 = a$ and $x_2 = b$ (with $y$ bound to $u$) is not a correct answer. Therefore, in the second phase, we propose a *base path criterion* to decide the soundness of an answer on the completed ABox. Towards a practical implementation, we materialize the base path criterion by base path precomputation and query rewriting.

Accordingly, we implemented a prototype to evaluate our approach using a relational database (DB2). Initially, both TBox and ABox are stored in an RDB. The ABox completion is conducted by running an RDB-based Datalog engine with our ABox completion rules. Finally, queries are rewritten and transformed to SQL, making them executable on top of the RDB. The experimental results show that our approach is practical for a large ABox with millions of triples.

Our key contributions in this paper are as follows: (a) we propose a practical approach for conjunctive query answering for acyclic $\mathcal{EL}^+$ that supports role inclusions; (b) we describe a low-cost implementation for this approach on top of a standard relational database management system; (3) we demonstrate its effectiveness and efficiency on a large-scale TBox and ABox.

We direct readers to the Appendix[1] for all proofs.

## 2 Preliminaries

### 2.1 $\mathcal{EL}^+$ and $\mathcal{EL}$ Family

In $\mathcal{EL}$, concept descriptions are $C ::= A | \exists R.C | C_1 \sqcap C_2 | \top$ where $\top$ is the top concept, $A$ is a concept name and $R$ is a role name. A TBox $\mathcal{T}$ is a finite set

[1] http://domino.research.ibm.com/comm/research_people.nsf/pages/jingmei.pubs.html/$FILE/iswc09_appendix.pdf

of general concept inclusions (GCIs) of the form $C_1 \sqsubseteq C_2$ where $C_1$ and $C_2$ are concept descriptions. An ABox $\mathcal{A}$ is a set of concept assertions $A(a)$ and role assertions $R(a_1, a_2)$ where $A$ is a concept name, $R$ is a role name, and $a, a_1, a_2$ are individual names. As usual in DLs, a knowledge base (KB) $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox. We use $NI$ (resp. $NC$ and $NR$) to denote the set of named individuals (resp. concept names and role names).

$\mathcal{EL}^+$ [3] extends $\mathcal{EL}$ by also allowing in the TBox a finite set of role inclusions of the form $R_1 \circ \cdots \circ R_k \sqsubseteq R_{k+1}$ where each $R_i$ is a role name and $1 \leqslant i \leqslant k+1$. Particularly important from the perspective of ontology applications, $\mathcal{EL}^+$ generalizes the following: role hierarchies $R \sqsubseteq S$ (aka. $\mathcal{ELH}$); transitive roles expressed by $R \circ R \sqsubseteq R$ (aka. $\mathcal{ELH}_{\mathcal{R}^+}$); and so-called left-identity rules $R \circ S \sqsubseteq S$ as well as right-identity rules $R \circ S \sqsubseteq R$. Finally, $\mathcal{EL}^{++}$ [1] is an extension of $\mathcal{EL}^+$ with the bottom concept $\bot$, the nominal $\{a\}$ and the concrete domain $p(f_1, \cdots, f_n)$.

As follows, we will focus on $\mathcal{EL}^+$ and its semantics is defined as usual. Moreover, $\mathcal{EL}^+$ TBoxes admit a normal form [3]. Any $\mathcal{EL}^+$ TBox in normal form consists of concept/role inclusions in one of the following forms: $A \sqsubseteq B, A_1 \sqcap A_2 \sqsubseteq B, A \sqsubseteq \exists R.B, \exists R.A \sqsubseteq B, R \sqsubseteq S$ and $R_1 \circ R_2 \sqsubseteq S$, where each $A, B, A_1, A_2$ is a concept name or the top concept $\top$, and each $R, S, R_1, R_2$ is a role name. Unless otherwise specified, this paper assumes that TBox $\mathcal{T}$ is in normal form.

A polynomial-time algorithm for TBox reasoning in $\mathcal{EL}^+$ has been proposed and implemented [3]. Using $C_1 \sqsubseteq_{\mathcal{T}} C_2$, we denote that $C_1$ is subsumed by $C_2$ w.r.t. $\mathcal{T}$. Given a TBox $\mathcal{T}$, we redefine cycles in $\mathcal{T}$ (to generalize the classical definition [4]). We say that $A$ *directly uses* $B$ in $\mathcal{T}$, if $A \sqsubseteq_{\mathcal{T}} \exists R.B$, and we call *uses* the transitive closure of the relation *directly uses*. Then $\mathcal{T}$ contains a cycle iff there exists a concept name in $\mathcal{T}$ that uses itself. Otherwise, $\mathcal{T}$ is called *acyclic*.

Finally, we also need to decide role subsumption in $\mathcal{EL}^+$. That is, given a TBox $\mathcal{T}$ and a finite sequence of role names $R_1, \cdots, R_k$ where $k \geqslant 3$, to find the set $\{R \mid R_1 \circ \cdots \circ R_k \sqsubseteq_{\mathcal{T}} R\}$. This problem can be computed in polynomial time in the size of $\mathcal{T}$. First, the role inclusions are expanded by exhaustively applying the following four rules: (1) if $R \sqsubseteq_{\mathcal{T}} S$ and $S \sqsubseteq_{\mathcal{T}} T$, then $R \sqsubseteq_{\mathcal{T}} T$; (2) if $R_1 \circ R_2 \sqsubseteq_{\mathcal{T}} S$ and $S \sqsubseteq_{\mathcal{T}} T$, then $R_1 \circ R_2 \sqsubseteq_{\mathcal{T}} T$; (3) if $R_1 \circ R_2 \sqsubseteq_{\mathcal{T}} S$ and $T \sqsubseteq_{\mathcal{T}} R_1$, then $T \circ R_2 \sqsubseteq_{\mathcal{T}} S$; (4) if $R_1 \circ R_2 \sqsubseteq_{\mathcal{T}} S$ and $T \sqsubseteq_{\mathcal{T}} R_2$, then $R_1 \circ T \sqsubseteq_{\mathcal{T}} S$. Then, given $R_1, \cdots, R_k$ where $k \geqslant 3$, we can recursively compute $\{S | R_1 \circ \cdots \circ R_i \sqsubseteq_{\mathcal{T}} S\}$ and $\{T | R_{i+1} \circ \cdots \circ R_k \sqsubseteq_{\mathcal{T}} T\}$, for $1 \leqslant i < k$, such that $\{R \mid S \circ T \sqsubseteq_{\mathcal{T}} R\}$.

## 2.2 Conjunctive Query Answering

Referring to [11,12], we introduce conjunctive queries and certain answers.

A conjunctive query $q$ is an expression of the form $\exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} = \{x_1, \cdots, x_m\}$ and $\boldsymbol{y} = \{y_1, \cdots, y_n\}$ are vectors of variables, while $\varphi$ is a conjunction of concept atoms $A(t)$ and role atoms $R(t_1, t_2)$. Variables in $\boldsymbol{x}$ are called distinguished variables (also called answer variables) and variables in $\boldsymbol{y}$ are non-distinguished (also called quantified variables). A term is a variable or a named individual, and we use $\mathrm{Term}(q)$ to denote the set of all terms in a query $q$.

Let $\mathcal{I}$ be an interpretation and $q = \exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$ a conjunctive query. A match for $\mathcal{I}$ and $q$ is a mapping $\pi : \mathrm{Term}(q) \to \Delta^{\mathcal{I}}$ such that: (1) $\pi(a) = a^{\mathcal{I}}$ for all named individuals $a \in \mathrm{Term}(q) \cap NI$; (2) $\pi(t) \in A^{\mathcal{I}}$ for all concept atoms $A(t) \in q$; (3) $(\pi(t_1), \pi(t_2)) \in R^{\mathcal{I}}$ for all role atoms $R(t_1, t_2) \in q$. If $\boldsymbol{x} = \{x_1, \cdots, x_m\}$ with $\pi(x_i) = a_i^{\mathcal{I}}$ for $1 \leqslant i \leqslant m$, then $\pi$ is called an $(a_1, \cdots, a_m)$-match for $\mathcal{I}$ and $q$. If such a match exists, we write $\mathcal{I} \vDash q[a_1, \cdots, a_m]$.

A certain answer for a conjunctive query $q = \exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$ and a knowledge base $\mathcal{K}$ is a tuple $[a_1, \cdots, a_m]$ such that, for any model $\mathcal{J}$ of $\mathcal{K}$, $\mathcal{J} \vDash q[a_1, \cdots, a_m]$. We use $cert(q, \mathcal{K})$ to denote the set of all certain answers for $q$ w.r.t $\mathcal{K}$.

## 3 ABox Completion

The purpose of this phase is to build a completed ABox by enriching the original ABox with inferred assertions, so that conjunctive queries can be answered on the completed ABox, instead of doing ABox reasoning at runtime.

### 3.1 ABox Completion Rules

To complete the ABox, one of the main difficulties is handling the existential concept inclusion $A \sqsubseteq \exists R.B$, which is a generative axiom in that it introduces new individuals not occurring in the original ABox.

A naive way to do the ABox completion is to compute: (1) a mapping $M_C$ from concepts to a subset of individuals; and (2) a mapping $M_R$ from roles to a binary relation on individuals. Below is the list of ABox completion rules.

$AR1$. If $A \sqsubseteq B \in \mathcal{T}$, $x \in M_C(A)$ and $x \notin M_C(B)$, then $M_C(B) := M_C(B) \cup \{x\}$

$AR2$. If $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}, x \in M_C(A_1)$, $x \in M_C(A_2)$ and $x \notin M_C(B)$, then $M_C(B) := M_C(B) \cup \{x\}$

$AR3$. If $\exists R.A \sqsubseteq B \in \mathcal{T}, (x, y) \in M_R(R)$, $y \in M_C(A)$ and $x \notin M_C(B)$, then $M_C(B) := M_C(B) \cup \{x\}$

$AR4$. If $R \sqsubseteq S \in \mathcal{T}$, $(x, y) \in M_R(R)$, then $M_R(S) := M_R(S) \cup \{(x, y)\}$

$AR5$. If $R_1 \circ R_2 \sqsubseteq S \in \mathcal{T}, (x, y) \in M_R(R_1)$, $(y, z) \in M_R(R_2)$, then $M_R(S) := M_R(S) \cup \{(x, z)\}$

$AR6$. If $A \sqsubseteq \exists R.B \in \mathcal{T}$, $x \in M_C(A)$, and there is no individual $y'$, s.t. $y' \in M_C(B)$ and $(x, y') \in M_R(R)$, then generate a new individual $y$, and $M_C(B) := M_C(B) \cup \{y\}$ and $M_R(R) := M_R(R) \cup \{(x, y)\}$

However, there will be a large number of individuals generated. Suppose $A_i \sqsubseteq \exists R.A_j$ with $A_0(a)$, where $A_i \neq A_j$ and $0 \leqslant i < j \leqslant m$. By applying $AR6$ to the named individual $a$, there are $2^m - 1$ individuals generated. Let $n$ be the number of $A_0$'s instances. The size of the final generated individuals will blow up to $n \times (2^m - 1)$.

To address this problem, an idea of "shared" individuals is adopted in this paper. Specifically, for each existential restriction $\exists R.B$, we will generate a fresh new individual (viz. *canonical individual*) $u$ with respect to $R$ and $B$, denoted by

$CI(u, R, B)$. To elaborate, for each existential concept inclusion $A \sqsubseteq \exists R.B$ and all instances $a \in M_C(A)$, we ensure that $u \in M_C(B)$ and $(a, u) \in M_R(R)$. Below is an alternate ABox completion rule $AR6'$ that replaces the previous $AR6$.

$AR6'$. If $A \sqsubseteq \exists R.B \in \mathcal{T}$, $x \in M_C(A)$ and $CI(u, R, B)$,
then $M_C(B) := M_C(B) \cup \{u\}$ and $M_R(R) := M_R(R) \cup \{(x, u)\}$

It is not hard to show that, by repeatedly applying rules $AR1 - 5$ and $AR6'$, the rule application procedure will eventually terminate, when no more changes to $M_C$ and $M_R$ occur. In fact, canonical individuals are those newly introduced anonymous individuals, and we have generated all canonical individuals as the preprocessing step for ABox completion, so that $M_C$ and $M_R$ are computed using at most $n \cdot m$ rule applications, yielding the data set bounded by $\mathbf{O}(n \cdot m)$, where $n$ and $m$ are linear in the size of $\mathcal{T}$ and $\mathcal{A}$.

Now, applying $AR6'$ to the above example, we will only generate $m$ individuals for each and every $\exists R.A_j$ where $1 \leqslant j \leqslant m$, even if there are $n$ instances of $A_0$. Besides, our ABox completion can benefit a lot from canonical individuals. First, it dramatically reduces the number of anonymous individuals to be generated. Second, it can be pre-computed for a given TBox, so as to reduce the cost of doing the ABox completion.

Thus, similar to the theoretical result presented in [11], our ABox completion can be done in linear time and yield a completed ABox whose size is linear w.r.t. both the TBox and the ABox.

## 3.2 ABox Completion Implementation

Our approach is implemented using an RDB system. Actually, via introducing canonical individuals, our ABox completion rules $AR1 - 5$ and $AR6'$ can be transformed into Datalog rules, and the completed ABox can be computed by running an RDB-based Datalog engine via a bottom-up evaluation strategy.

First, we design a database schema s.t. the normalized TBox is stored in 6 tables, namely, `atomicSub`, `existsSub`, `gciInter`, `gciExists`, `subRole` and `roleChain`. These tables correspond to the TBox axioms in form of: $A \sqsubseteq B, A \sqsubseteq \exists R.B, A_1 \sqcap A_2 \sqsubseteq B, \exists R.A \sqsubseteq B, R \sqsubseteq S$ and $R_1 \circ R_2 \sqsubseteq S$, respectively. Similarly, the ABox is stored in two tables, `typeOf` and `roleStmt`, corresponding to assertions of $A(x)$ and $R(x, y)$, respectively. The pre-computed canonical individuals are stored in the table `canonInd`.

Second, we translate the ABox completion rules $AR1 - 5$ and $AR6'$ into Datalog rules, in addition to two initialization rules s.t., (1) if `typeOf`$(x, y)$ then `infTypeOf`$(x, y)$; (2) if `roleStmt`$(R, x, y)$ then `infRoleStmt`$(R, x, y, 0)$. Now, running an RDB-based Datalog engine with the rules, and via a bottom-up evaluation strategy, all inferred assertions will be incrementally stored in tables `infTypeOf` and `infRoleStmt`. In particular, the last column in `infRoleStmt` is used to indicate the origin of inferred role assertions (the reason for doing that will be explained in the next section). Specifically, if $R(x, y)$ is inferred by $AR6'$, then we store it as `infRoleStmt`$(R, x, y, 1)$; and if $R(x, y)$ is inferred by $AR4 - 5$, then we store it as `infRoleStmt`$(R, x, y, 2)$. We remark that rules of

$AR4-5$ and $AR6'$ are allowed to fire, even if the role assertion $R(x,y)$ that they are attempting to infer has already been computed as $(x,y) \in M_R(R)$. For instance, suppose $\mathcal{T} = \{A \sqsubseteq \exists R_1.B, B \sqsubseteq \exists R_2.C, A \sqsubseteq \exists R_2.C, R_1 \circ R_2 \sqsubseteq R_2\}$ and $\mathcal{A} = \{A(a)\}$. After the rule execution, we have `canonInd`$(u_1, R_1, B)$ and `canonInd`$(u_2, R_2, C)$, in addition to `infRoleStmt`$(R_2, a, u_2, 1)$ because of $A \sqsubseteq \exists R_2.C$, and `infRoleStmt`$(R_2, a, u_2, 2)$ because of $R_1 \circ R_2 \sqsubseteq R_2$.

Thus, the ABox completion is done as described above, and we use $\mathcal{A}_\alpha$ to denote the completed ABox, where the set of canonical individuals in $\mathcal{A}_\alpha$ is denoted by $CI$.

As follows, we define an interpretation $\mathcal{I}_\alpha$ w.r.t. $\mathcal{A}_\alpha$ s.t.

- $\Delta^{\mathcal{I}_\alpha} := NI \cup CI$
- $a^{\mathcal{I}_\alpha} := a$ for all $a \in NI$
- $A^{\mathcal{I}_\alpha} := \{e \in \Delta^{\mathcal{I}_\alpha} \mid A(e) \in \mathcal{A}_\alpha\}$ for all $A \in NC$
- $R^{\mathcal{I}_\alpha} := \{(e_1,e_2) \in \Delta^{\mathcal{I}_\alpha} \times \Delta^{\mathcal{I}_\alpha} \mid R(e_1,e_2) \in \mathcal{A}_\alpha\}$ for all $R \in NR$

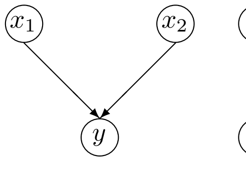
It is not hard to show that $\mathcal{I}_\alpha$ is a model of $\mathcal{K}$, and if $(a_1, \cdots, a_m) \in cert(q, \mathcal{K})$ then $\mathcal{I}_\alpha \models q[a_1, \cdots, a_m]$, but not vice versa.

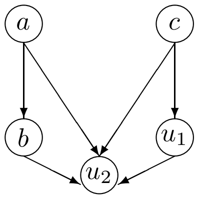
## 4 Query Answering

After ABox completion, queries are ready for answering. However, problems arise due to the introduction of canonical individuals. In this section, we will illustrate a running example, followed by our solution and optimization.

### 4.1 The Running Example

We use a query $q = \exists y.R(x_1,y) \wedge R(x_2,y)$ and an $\mathcal{EL}^+$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ as the running example, where $\mathcal{T} = \{C \sqsubseteq \exists R_2.B, B \sqsubseteq \exists R.D, R_1 \circ R \sqsubseteq R_3, R_2 \circ R \sqsubseteq R_4, R_3 \sqsubseteq R, R_4 \sqsubseteq R\}$ and $\mathcal{A} = \{B(b), C(c), R_1(a,b)\}$. Note that $\mathcal{T}$ is irregular. The query is shown below. By the ABox completion, there are two canonical individuals generated, i.e. $u_1$ for $\exists R_2.B$ and $u_2$ for $\exists R.D$, in addition, $\mathcal{A}_\alpha = \mathcal{A} \cup \{B(u_1), D(u_2), R(b,u_2), R_2(c,u_1), R(u_1,u_2), R_3(a,u_2), R(a,u_2), R_4(c,u_2), R(c,u_2)\}$, as sketched below. Now, matching the query $q$ with the completed ABox $\mathcal{A}_\alpha$, we have the answer set, i.e., $\{(a,a),(b,b),(c,c),(a,b),(b,a),(b,c),(c,b),(a,c),(c,a)\}$. As shown below, the last four ones, $(b,c),(c,b),(a,c),(c,a)$, are incorrect answers.


$q = \exists y.R(x_1,y) \wedge R(x_2,y)$


$\mathcal{A}_\alpha$

| $x_1$ | $x_2$ | $y$ | answer |
|---|---|---|---|
| $a$ | $a$ | $u_2$ | correct |
| $b$ | $b$ | $u_2$ | correct |
| $c$ | $c$ | $u_2$ | correct |
| $a$ | $b$ | $u_2$ | correct |
| $b$ | $a$ | $u_2$ | correct |
| $b$ | $c$ | $u_2$ | incorrect |
| $c$ | $b$ | $u_2$ | incorrect |
| $a$ | $c$ | $u_2$ | incorrect |
| $c$ | $a$ | $u_2$ | incorrect |

We observe that, first, there is a reverse tree in the query. Intuitively, a role atom $R(x, y)$ can be regarded as an edge with the direction pointing from $x$ to $y$. Roughly, if there are multiple edges pointing to the same $y$, we call it a *reverse tree*. For $q$, role atoms $R(x_1, y)$ and $R(x_2, y)$ are regarded as two edges pointing to the same $y$, which makes this a reverse tree. Second, the completed data $R(a, u_2)$ and $R(c, u_2)$ matches the reverse tree incorrectly. This is due to the canonical individual $u_2$ that was generated for $\exists R.D$ and shared by $b$ and $u_1$, due to $B \sqsubseteq \exists R.D$. Furthermore, the "sharing" is extended to $a$ and $c$, due to $R_1 \circ R \sqsubseteq_{\mathcal{T}} R$ and $R_2 \circ R \sqsubseteq_{\mathcal{T}} R$, respectively, where $R_1(a, b)$ and $R_2(c, u_1)$. As proposed in [11,12], to guarantee sound answers, the query would be rewritten as $q^* = \exists y.R(x_1, y) \wedge R(x_2, y) \wedge (CI(y) \rightarrow x_1 = x_2)$ s.t., if $y$ is matched with a canonical individual, then $x_1$ and $x_2$ need to be matched with the same individual. In this way, unsound answers like $(x_1/a, x_2/c)$ are filtered. However, answers are now incomplete, because some correct answers such as $(x_1/a, x_2/b)$ are also removed. The reason why the existing strategy of query rewriting fails in this case is due to the increased role expressivity. A big difference between $(x_1/a, x_2/c)$ and $(x_1/a, x_2/b)$ is that $a$ and $b$ share $u_2$ in a native way, since $R_1 \circ R \sqsubseteq_{\mathcal{T}} R$.

Therefore, we need to distinguish variable bindings considering reverse trees in the query. To this end, we first need to distinguish different role assertions in the completed ABox, some of which are generated by existential concept inclusions, and some of which are derived from role inclusions. The former is the origin of incorrect query answering, while the latter extends the incorrectness. Recalling our ABox completion implementation, we mark a tag in `infRoleStmt`, making $R(b, u_2), R_2(c, u_1), R(u_1, u_2)$ tagged by 1 and $R_3(a, u_2), R(a, u_2), R_4(c, u_2)$, $R(c, u_2)$ tagged by 2. Thus, we distinguish role assertions tagged by 1 from those tagged by 2, which are "purely" derived by $AR6'$ and are the root cause of the problem. We call them *base triples*.

Furthermore, we build so-called *base paths* by traversing the graph of base triples. In our example, there are three base triples, i.e., $R(b, u_2), R_2(c, u_1)$, $R(u_1, u_2)$, and three base paths, i.e., $bRu_2, cR_2u_1, cR_2u_1Ru_2$. Given a base path, the first individual is called the *head*, and the last individual is called the *tail*, while all individuals in the base path are said to be *directly located* in it. If there is a named individual reachable to the head of a base path, we refer to it as being *indirectly located* in the base path. For instance, $a$ is reachable to $b$, making $a$ indirectly located in $bRu_2$.

These base paths are now used for filtering incorrect answers. Informally, all terms occurring in a reverse tree of the query are required to be (in)directly located in a single base path, so called the *base path criterion*. In our example, $a$ and $b$ are, respectively, indirectly and directly located in $bRu_2$, so $(x_1/a, x_2/b)$ is a correct answer. Conversely, there is no base path making both $a$ and $c$ (in)directly located, so $(x_1/a, x_2/c)$ is not a correct answer.

Meanwhile, we realize that it is infeasible to check base paths at runtime during query answering. As illustrated above, there are at least 9 matches for $\mathcal{A}_\alpha$ and $q$ in our example. To reduce the cost, we precompute and materialize all possible role assertions in the same base path in the table `BPath(path, tail, node1, node2, role)`, where the `path` has the `tail`, and `node1` is (in)directly located, while `node2` is directly located in the `path` via the `role` relationship. Below is the BPath storage of our example. For example, BPath$(bRu_2, u_2, a, u_2, R)$ indicates the path $bRu_2$ has the tail $u_2$, and $a$ is indirectly located while $u_2$ is directly located via the assertion $R(a, u_2)$.

**Table 1.** An example of the BPath storage

| `path` | `tail` | `node1` | `node2` | `role` |
|---|---|---|---|---|
| $cR_2u_1$ | $u_1$ | $c$ | $u_1$ | $R_2$ |
| $bRu_2$ | $u_2$ | $b$ | $u_2$ | $R$ |
| $bRu_2$ | $u_2$ | $a$ | $u_2$ | $R_3$ |
| $bRu_2$ | $u_2$ | $a$ | $u_2$ | $R$ |
| $cR_2u_1Ru_2$ | $u_2$ | $c$ | $u_1$ | $R_2$ |
| $cR_2u_1Ru_2$ | $u_2$ | $u_1$ | $u_2$ | $R$ |
| $cR_2u_1Ru_2$ | $u_2$ | $c$ | $u_2$ | $R_4$ |
| $cR_2u_1Ru_2$ | $u_2$ | $c$ | $u_2$ | $R$ |

Finally, our base path criterion is applicable for query rewriting. For $q$, we rewrite it as $q^* = \exists y.R(x_1, y) \wedge R(x_2, y) \wedge (CI(y) \rightarrow \exists p.BPath(p, y, x_1, y, R) \wedge BPath(p, y, x_2, y, R))$. Here, the satisfiability of $CI(y)$ means $y$'s match is a canonical individual, while the satisfiability of both $BPath(p, y, x_1, y, R)$ and $BPath(p, y, x_2, y, R)$ means the matches of $x_1$ and $x_2$ are (in)directly located in a single base path $p$. As shown in Table 1, both BPath$(bRu_2, u_2, a, u_2, R)$ and BPath$(bRu_2, u_2, b, u_2, R)$ are satisfied, making $(x_1/a, x_2/b)$ a correct answer.

In the next three subsections, we will formally present our solution, including how to identify reverse tress in the query, how to compute base paths in the ABox completion, and how to rewrite the query according to our base path criterion.

## 4.2 Reverse Trees in Query

Before we provide a formal definition of reverse trees, we remark that little attention is paid to multiple edges pointing to the same $y$, where $y$ is bound to a named individual. This is because only matches sharing some canonical individuals are problematic. Moreover, $\mathcal{EL}^+$ models are split [11], i.e., there is no role assertion pointing from an unnamed individual to a named one.

Formally, let $q = \exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$ be a conjunctive query. We use $GT_q \subseteq \text{Term}(q)$ to denote the set of grounded terms in $q$, s.t. for each $t \in \text{Term}(q)$: (1) if $t \in NI \cup \boldsymbol{x}$ then $t \in GT_q$; (2) if $R(t, t') \in q$ and $t' \in GT_q$ then $t \in GT_q$.

Next, we use $\gamma_q$ to denote the set of reverse tree roots in the query $q$, i.e., $\gamma_q := \{y \notin GT_q | \sharp\{R(x, y) \in q\} > 1\}$, where $\sharp$ denotes the cardinality.

Now, for each reverse tree root $y \in \gamma_q$, we compute $E_y^{(0)} := \{R(x, y) \in q\}$ and $E_y^{(i+1)} := E_y^{(i)} \cup \{R(s, t) \in q \mid t \in \{s' | R'(s', t') \in E_y^{(i)}\}$ and $t \notin GT_q\}$. The computation terminates, when $E_y^{(i)} = E_y^{(i+1)}$. $E_y^{(0)}$ refers to the set of multiple edges pointing to the same $y$. By incrementally expanding in a reverse direction, $E_y^{(i)}$ contains additional edges, all of which point to non-grounded terms, until we reach a fixedpoint. The resultant set $E_y := \bigcup_{i \geqslant 0} E_y^{(i)}$, and we use $E_y$ to denote the set of all reverse tree edges rooted by $y \in \gamma_q$.

Finally, we use $RT_q$ to denote the set of reverse trees, where each reverse tree is a pair $(y, E_y)$ with $y \in \gamma_q$ and there is no pair $(y', E_{y'})$ in $RT_q$ such that $E_y \subset E_{y'}$. We call a query $q$ free of reverse trees, if $RT_q$ is empty.

It is not hard to verify that $GT_q$ and $RT_q$ can be computed in time polynomial in the size of $q$. Moreover, the size of $GT_q$ is linear in the size of $q$, and the size of $RT_q$ is polynomial in the size of $q$.

As shown in the following theorem, for queries free of reverse trees, query answering over the completed ABox is sound and complete, even in cyclic $\mathcal{EL}^+$ KBs. Thus, if the application requires only queries free of reverse trees, the rest of this section can be skipped over.

**Theorem 1.** *$cert(q, \mathcal{K}) = ans(q, \mathcal{A}_\alpha)$, given that $q$ is a conjunctive query free of reverse trees and $\mathcal{K}$ is an $\mathcal{EL}^+$ knowledge base.*

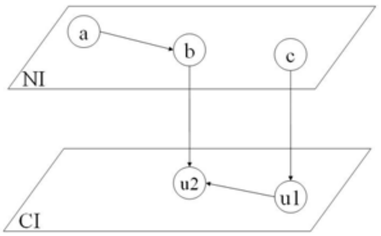
To see which queries are free of reverse trees, consider the following samples. First, $q_1 = \exists y_1, y_2.R_1(x_1, y_1) \wedge R_2(x_2, y_2)$ is forest-shaped (a notion well defined in DL literature [8]), and we claim that forest-shaped queries are all free of reverse trees. Next, neither $q_2 = \exists y_1, y_2, y_3.R(y_1, y_2) \wedge R(y_2, y_3) \wedge R(y_3, y_1)$ nor $q_3 = \exists y_1, y_2.R_1(x_1, y_1) \wedge R_2(x_2, y_1) \wedge R_3(y_1, y_2) \wedge R_4(y_2, x_3)$ is forest-shaped, while both $q_2$ and $q_3$ are free of reverse trees. In fact, there is no multiple edge in $q_2$, which makes it free of a reverse tree. In $q_3$, the multiple edges point to $y_1$ which is a grounded term due to the answer variable $x_3$, so there is no reverse tree root and $q_3$ is also free of a reverse tree.

## 4.3 Base Paths in ABox Completion

First, we need to identify base triples in the ABox completion. Taking advantage of the last column in `infRoleStmt`, $BT$ is used to denote the set of base triples s.t. $BT := \{R(x, y) | \texttt{infRoleStmt}(R, x, y, 1)\} \setminus \{R(x, y) | \texttt{infRoleStmt}(R, x, y, 2)\}$. In other words, $BT$ consists of triples which are "purely" derived by $AR6'$.

Next, we traverse the graph of $BT$ to compute base paths. Given a canonical individual $u \in CI$, we use $BP(u)$ to denote the set of base paths ending up with $u$ such that $BP(u) := \{u_0 R_1 u_1 \cdots R_k u_k | u_0 \in NI, u_k = u, u_i \in CI, R(u_{i-1}, u_i) \in BT, 1 \leqslant i \leqslant k\}$. The following figure illustrates the intuition of our definition.

There are two planes: the NI plane contains all triples between named individuals, and the CI plane contains only the base triples between canonical individuals. Meanwhile, base triples from named individuals to canonical individuals are those links from the NI plane to the CI plane. Here is the running example.



Intuitively, a base path starts from a named individual (i.e. the head), followed by a base triple jumping from the NI plane to the CI plane, and after traversal, it ends up with a canonical individual (i.e. the tail). Therefore, all individuals in a base path are canonical ones, except the head. Named individuals indirectly located in a base path are those reachable to its head in the NI plane.

It is not hard to show that, disregarding cyclic $\mathcal{EL}^+$ KBs, the base path computation will terminate, and using depth-first-search or breadth-first-search, its time complexity is proportional to the number of nodes plus the number of edges in the graph they traverse, i.e., $\mathbf{O}(|NI|+|CI|+|BT|)$. In other words, $BP$ can be computed in time linear in the size of $\mathcal{A}_\alpha$. Unfortunately, in the worst-case, the size of base paths is exponential in the size of TBox and polynomial in the size of ABox. We believe this upper bound is unlikely to occur in practice, and Section 4.5 presents our optimization techniques to address this issue.

Finally, we precompute and materialize all possible role assertions in the same base path in the table `BPath(path, tail, node1, node2, role)`. For a base path $p$: $u_0R_1u_1\cdots R_ku_k$, we store it as $\mathrm{BPath}(p, u_k, u_i, u_j, R)$, where $0 \leqslant i < j \leqslant k$ and $R_{i+1} \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$, as well, $\mathrm{BPath}(p, u_k, a_0, u_j, R)$ where $0 < j \leqslant k$ and $R_0 \circ R_1 \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$ with $R_0(a_0, u_0) \in \mathcal{A}_\alpha$.

Here, we remark two key points about the `BPath` storage. One is how to compute the `role`, which denotes the super roles for the role chain from `node1` to `node2`. For $\mathrm{BPath}(p, u_k, u_i, u_j, R)$ where $0 \leqslant i < j \leqslant k$, we need to compute the set $\{R \mid R_{i+1} \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R\}$, as discussed in the Section 2 (Preliminaries). Similarly, for $\mathrm{BPath}(p, u_k, a_0, u_j, R)$, where $0 < j \leqslant k$, we compute the set $\{R \mid R_0 \circ R_1 \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R\}$. Another is how to compute all named individuals reachable to $u_0$. Suppose $a_0$ is reachable to $u_0$, via a sequence of named individuals $a_1, \cdots, a_n$ and a sequence of role names $S_1, \cdots, S_n$, where $a_n = u_0$ and $S_{i+1}(a_i, a_{i+1}) \in \mathcal{A}_\alpha$ for any $0 \leqslant i < n$, given $S_1 \circ \cdots \circ S_n \circ R_1 \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$. Thanks to the TBox normalization [1], a role inclusion in form of $R_1 \circ \cdots \circ R_k \sqsubseteq R_{k+1}$ will be normalized inductively by $R_1 \circ \cdots \circ R_{k-1} \sqsubseteq R'_k$ and $R'_k \circ R_k \sqsubseteq R_{k+1}$, where $R'_k$ is a newly introduced role name. In this respect, we are convinced that, given $S_1 \circ \cdots \circ S_n \circ R_1 \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$, there is a role name $R_0$ (introduced by the normalization) s.t. $S_1 \circ \cdots \circ S_n \sqsubseteq_\mathcal{T} R_0$ and $R_0 \circ R_1 \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$. Thus, with $u_0$ pinpointed, the set of named individuals reachable to $u_0$ are $\{a_0 \mid R_0(a_0, u_0) \in \mathcal{A}_\alpha\}$.

## 4.4 Query Rewriting

Below, we define a base path criterion, then rewrite the query according to it.

**Definition 1 (Base Path Criterion).** *Let $\tau$ be a match for $\mathcal{I}_\alpha$ and $q$.*

*We say that a reverse tree $(y, E_y) \in ST_q$ is satisfied by $\tau$ if, $\tau(y) \in CI$ implies there is a base path $u_0R_1u_1\cdots R_ku_k \in BP(\tau(y))$, and for any $R(s,t) \in E_y$, whenever $\tau(t) = u_j$ and $1 \leqslant j \leqslant k$,*

- *$\tau(s) = u_i$ and $0 \leqslant i < j$, given $R_{i+1} \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$, otherwise;*
- *$\tau(s) = a_0$ and $0 \leqslant i < n$, $(a_i, a_{i+1}) \in S_{i+1}^{\mathcal{I}_\alpha}$ with $a_n = u_0$, given $S_1 \circ \cdots \circ S_n \circ R_1 \circ \cdots \circ R_j \sqsubseteq_\mathcal{T} R$.*

*The base path criterion is that all reverse trees of $q$ are satisfied by the match $\tau$ for $\mathcal{I}_\alpha$ and $q$.*

Since we have stored all base paths, the base path criterion can be directly adopted by query rewriting. The rewritten query $q^*$ is defined as

$$q^* := q \wedge \bigwedge_{(y,E_y)\in RT_q} (CI(y) \to \exists p. \bigwedge_{R(s,t)\in E_y} (CI(t) \to BPath(p,y,s,t,R)))$$

and the size of rewritten queries is bounded by $\mathbf{O}(|q|^2)$.

By definition, an $(a_1,\cdots,a_m)$-match $\tau$ for $\mathcal{I}_\alpha$ and $q$ satisfies a reverse tree $(y,E_y)\in ST_q$ if, $\tau(y)\in CI$ implies there is $p$ such that, whenever $\tau(t)\in CI$, we have $\text{BPath}(\tau(p),\tau(y),\tau(s),\tau(t),R)$, for any $R(s,t)\in E_y$. If there exists such a match satisfying all reverse trees, we write $\mathcal{I}_\alpha \cup \text{BPath} \vDash q^*[a_1,\cdots,a_m]$.

Similar to [11,12], we use $ans(q,\mathcal{A}_\alpha)$ to denote the set of answers that a relational database system returns for $q$ over $\mathcal{A}_\alpha$, while $ans(q^*,\mathcal{A}_\alpha\cup BPath)$ for $q^*$ over $\mathcal{A}_\alpha$ with BPath. This paper contributes the following theorem.

**Theorem 2.** *$cert(q,\mathcal{K}) = ans(q^*,\mathcal{A}_\alpha \cup BPath)$, given that $q$ is a conjunctive query and $\mathcal{K}$ is an acyclic $\mathcal{EL}^+$ knowledge base.*

Interestingly, if $\mathcal{K}$ is an $\mathcal{EL}$ knowledge base, then our base path criterion is reducible to the approach proposed for $\mathcal{EL}$ [11]. In fact, the Definition 1 will be simplified s.t., for any $R(s,t)\in E_y$, if $\tau(t)=u_j$ then $\tau(s)=u_{j-1}$, where $1\leqslant j\leqslant k$. Suppose $s_1,\cdots,s_l$ be a sequence of terms and all $R(s_i,t)\in E_y$ for $1\leqslant i\leqslant l$. Now, if $\tau(t)=u_j$ then $\tau(s_1)=\cdots=\tau(s_l)=u_{j-1}$, and in the rewritten query, it becomes $CI(t)\to(s_1=s_2\wedge\cdots\wedge s_{l-1}=s_l)$.

## 4.5 Optimization

Observing the possibility of using our approach for $\mathcal{EL}$, we consider removing some BPath records which contribute little to the *plus* part of $\mathcal{EL}^+$, towards an optimal storage. Formally, we define the set $NR^+ := \{R|R_1\circ R_2 \sqsubseteq_\mathcal{T} R_3, R\sqsubseteq_\mathcal{T} R_i, 1\leqslant i\leqslant 3\}$. Our strategy is to check the BPath storage, and for any $\text{BPath}(p,z,x,y,R)$, if $R\notin NR^+$, then this BPath record can be removed.

Correspondingly, we also need to redefine the rewritten query $\bar{q}^*$ as follows.

$$\bar{q}^* := q \wedge \bigwedge_{(y,E_y)\in RT_q} (CI(y) \to \exists p. \bigwedge_{R(s,t)\in E_y, R\in NR^+} (CI(t) \to BPath(p,y,s,t,R))$$

$$\wedge \bigwedge_{(\{t_1,\cdots,t_l\},\zeta)\in \text{Fork}^y_=} (CI(t_\zeta) \to \bigwedge_{1\leqslant i<l} t_i = t_{i+1}) \wedge \bigwedge_{(I,\zeta)\in\text{Fork}^y_\mathcal{H}} (CI(t_\zeta) \to \bigvee_{R\in I} R(t^{\text{pre}}_\zeta, t_\zeta)))$$

The definitions of $\text{Fork}^y_=$ and $\text{Fork}^y_\mathcal{H}$ are almost the same as those defined for $\mathcal{ELH}^{dr}_\bot$ [12]. The main differences are $\text{pre}(\zeta) := \{t|R(t,t')\in E_y$ for some $R\notin NR^+$ and $t'\in\zeta\}$ and $\text{in}(\zeta) = \{R|R(t,t')\in E_y$ for some $R\notin NR^+$ and $t'\in\zeta\}$. Here, $R(t,t')\in E_y$ and $R\notin NR^+$ are imposed in our context.

To see why our optimization retains the correctness, consider the query $q = \exists y_1, y_2, y.R_1(x_1, y_1) \wedge R_2(x_2, y_1) \wedge R(y_1, y) \wedge R(y_2, y)$ over an $\mathcal{EL}^+$ KB with $R_1 \sqsubseteq R_2$ and $R \circ R \sqsubseteq R$. There is one and only one reverse tree $(y, E_y) \in RT_q$, where $E_y$ consists of all role atoms in $q$, and $NR^+ = \{R\}$. The non-optimized rewritten query $q^*$ is $q \wedge (CI(y) \rightarrow \exists p.BPath(p, y, y_1, y, R) \wedge BPath(p, y, y_2, y, R) \wedge (CI(y_1) \rightarrow BPath(p, y, x_1, y_1, R_1) \wedge BPath(p, y, x_2, y_1, R_2)))$, and the optimized rewritten query $\bar{q}^*$ is $q \wedge (CI(y) \rightarrow \exists p.BPath(p, y, y_1, y, R) \wedge BPath(p, y, y_2, y, R)$ $\wedge\ (CI(y_1) \rightarrow x_1 = x_2) \wedge (CI(y_1) \rightarrow R_1(x_2, y_1))$. Below shows the theorem for optimization, where $\overline{\text{BPath}}$ denotes the BPath storage after removal.

**Theorem 3.** *$cert(q, \mathcal{K}) = ans(\bar{q}^*, \mathcal{A}_\alpha \cup \overline{BPath})$, given that $q$ is a conjunctive query and $\mathcal{K}$ is an acyclic $\mathcal{EL}^+$ knowledge base.*

This section is now concluded with the theorem for complexity and data size.

**Theorem 4.** *Conjunctive query answering on acyclic $\mathcal{EL}^+$ knowledge base can be done in time polynomial in the size of the query, the TBox and the ABox, with data of a polynomial-size blowup w.r.t. the query and the ABox while of an exponential-size blowup w.r.t. the TBox.*

## 5 Preliminary Experiments

We have implemented a prototype to evaluate our approach using Java SDK 1.5 and DB2 V9.3 Enterprise Edition. It performs three steps: (1) does ABox completion with canonical individuals and stores the completed ABox data in the RDB; (2)computes the base paths and stores them in the RDB as well; (3) rewrites the query according to the base path criteria, transforms the query into SQL and executes it on the completed ABox with base paths. The preliminary experiments are performed on an X-3650 server with 8G memory and two 3.0GHz Xeon CPUs.

**TBox.** The SNOMED CT ontology is generated by executing the perl script provided in the SNOMED CT 2009Jan release package, on the SNOMED CT 2007Jan release content files (because our data uses the 2007Jan version). It is formulated as an acyclic $\mathcal{EL}^+$ ontology that contains 308,832 concept names, 62 role names with 47,317 concept definitions and 261,514 primitive concept definitions. It also contains 11 role subsumption axioms and one right identity axiom: $caustiveAgent \circ hasActiveIngredient \sqsubseteq caustiveAgent$ [2]. After normalization, the TBox includes 463,971 concept names, 576,971 axioms of form $A \sqsubseteq B$, 428,905 axioms of form $A \sqsubseteq \exists R.B$, 49,454 axioms of form $\exists R.A \sqsubseteq B$ and 118,124 axioms of form $A_1 \sqcap A_2 \sqsubseteq B$.

[2] Actually, the stated right identity axiom in the SNOMED CT 2009Jan Release is $directSubstance \circ hasActiveIngredient \sqsubseteq directSubstance$. We do not use this axioms because there is no concept B appeared in both $A \sqsubseteq \exists directSubstance.B$ and $B \sqsubseteq \exists hasActiveIngredient.C$ in the 2007Jan Release.

**ABox.** The ABox of SNOMED CT ontology is constructed partly from a collection of 100,000 HL7 Clinical Document Architecture (CDA)[7] documents collected from a large hospital in southern China. CDA is a widely adopted standard to represent the electronic clinical documents, such as clinical notes and prescriptions. These CDA documents are CDA level 3 documents with clinical statements using codes from SNOMED CT 2007-Jan release.

We developed an XML-2-RDF transformer that can extract RDF triples from the structured body part of CDA documents. For each CDA document, the XML-2-RDF transformer outputs a single RDF document.In addition, because the collected CDA documents do not cover the concepts and roles involved in the right identity axiom, we generate additional instances randomly for the classes and roles involved in the right identity axiom, and then merge them into the ABox. In our experiment, we generate on average 30 instances per CDA document. The transformed RDF documents plus the randomly generated instances are partitioned into four data sets (ABoxes) according to the number of CDA documents, i.e, 0.1K, 1K, 10K and 100K.

**ABox Data completion.** After loading the normalized SNOMED CT ontology and the ABox into the RDB, we use the RDB-based datalog engine developed as part of the IBM SHER framework to perform ABox completion. To evaluate the benefits of canonical individual, we implemented both the intuitive approach and the canonical-individual-based approach.

In the intuitive approach, the algorithm first evaluates the ABox datalog rules ($AR$1-5) by a semi-naive approach, then iteratively does the following steps until no more new facts are generated: (1)For the TBox axiom: $A \sqsubseteq \exists R.B$, and ABox assertion $A(a)$, add new facts $R(a,u), B(u)$ into the ABox if there is no individual $w$ satisfying both $R(a,w)$ and $B(w)$; (2) Incrementally evaluate the ABox completion rules with the new facts.

In the canonical-individual-based approach, the algorithm first generates canonical individuals for existential restrictions, then evaluates the ABox datalog rules $AR$1-5,$AR6'$ via a semi-naive evaluation. In this way, the overall number of generated anonymous (canonical) individuals is fixed – 76,614 in our case.

Table 2 summarizes the ABox completion results for both approaches. For each ABox, we report the number of individuals (IND), the numbers of triples (TRP) in the original ABox and in the completed ABox in both approaches, and the data completion time (excluding the ABox data loading time).

**Table 2.** ABox data completion results

| Dataset | Original | | Intuitive Approach | | | CI-based Approach | | |
|---|---|---|---|---|---|---|---|---|
| | IND | TRP | IND | TRP | Time | IND | TRP | Time |
| 0.1K | 16K | 22K | 77K | 1,018K | 4.3h | 16K | 310K | 53s |
| 1K | 169K | 227K | 793K | 10,395K | 58.4h | 172K | 4,099K | 0.3h |
| 10K | 1,614K | 2,155K | N/A | N/A | N/A | 1,618K | 39,923K | 1.4h |
| 100K | 15,097K | 22,157K | N/A | N/A | N/A | 15,102K | 387,316K | 13.9h |

As can be seen from the Table 2, in the intuitive approach, the completed ABox has a polynomial-size blowup. Also, it is very time-consuming to generate due to many iterative calls to the datalog engine. When using canonical individuals, the number of newly added individuals in the completed ABox is always less than a constant (76,614 in our case). The size of inferred triples is about 10-20 times that of the original triples because of the deep hierarchy of the normalized SNOMED CT ontology. For example, the concept "headache" has 15 super concepts in original ontology, but has 27 super concepts in the normalized ontology. Our experiment confirms that it is necessary to introduce canonical individual for ABox data completion in practice.

**Base path generation and query performance.** After performing the ABox completion, the base path can be generated and the query can be rewritten and executed on the completed ABox with base paths. We first compare the efficiency of base path generation of the initial approach and the optimized approach. Then we design three typical queries to test the query performances. These queries are designed according to the three typical query shape: point, tree and graph.

Q1(x)= $Headache(x)$

Q2(x) =$\exists y\ \ HearingNormal(x), findingSite(x, y), EntireLeftEar(y)$

Q3(x,y)=$\exists z\ \ causativeAgent(x, z), causativeAgent(y, z)$

In our implementation, the (rewritten) conjunctive query is represented using the RDF SPARQL query language. The SPARQL query is further transformed to SQL by a RDF-2-RDB mapping engine.

Table 3 summarizes the base path generation and query performance results for each ABox. We report the count of base path records (BPSize), the time to generate them (GenTime) and the query execution time for Q3 using the initial approach and the optimized approach. Since Q1 and Q2 are free of reverse trees, the base path optimization has no effect on their performances.

**Table 3.** Base path generation and query performance results

| Dataset | without optimization | | | with optimization | | | Q1 | Q2 |
|---|---|---|---|---|---|---|---|---|
| | BPSize | GenTime | Q3 | BPSize | GenTime | Q3 | | |
| 0.1K | 172K | 480s | 321ms | 10K | 36.6s | 50ms | 10ms | 30ms |
| 1K | 1,917K | 2,237s | 3.5s | 199K | 228s | 2.0s | 61ms | 500ms |
| 10K | 16,764K | 4.2h | 22.1s | 1,812K | 1.1h | 12.7s | 83ms | 1.6s |
| 100K | N/A | N/A | N/A | 18,734K | 7.8h | 313s | 712ms | 14.6s |

Our experiment is still at a preliminary stage due to the limitations of the ontology and data. However, from the results, we can observe that: (1) the base path optimization dramatically reduces the number of stored base paths in this case, because only five of SNOMED CT's 62 roles appear in the "real" role inclusions, and further improves the performance of the rewritten query Q3 because self-joins on the base path table are involved; (2) the query answering is scalable to a large data set because it can leverage the underlying relational database for scalability.

## 6 Related Work

In recent years, a rich literature on the subject of conjunctive query answering on the $\mathcal{EL}$ family has emerged, including both theoretical and practical work.

Theoretically, there are undecidable problems identified, such as answering conjunctive queries in both $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$ [14]. Meanwhile, related work [10] presents that CQ answering on regular $\mathcal{EL}^{++}$ is decidable. However, we observed that the notion of regularity syntactically depends on the normal form of TBox. In other words, it can happen that a TBox becomes (ir)regular by normalizing it differently, even if this does not change the semantics. For example, suppose $R_1 \circ R_2 \circ R_3 \sqsubseteq R_3$ be a role inclusion originally asserted in the TBox. On the one hand, this TBox can be normalized as $R_1 \circ R_2 \sqsubseteq R'$ and $R' \circ R_3 \sqsubseteq R_3$, which appear as regular. On the other hand, this TBox can also be normalized as $R_2 \circ R_3 \sqsubseteq R''$ and $R_1 \circ R'' \sqsubseteq R_3$, which becomes irregular. Besides, our acyclic $\mathcal{EL}^+$ TBoxes admit the irregularity. Disregarding the rich expressivity of roles, CQ answering on $\mathcal{EL}$ and $\mathcal{ELH}$ has been proved decidable and PTIME-complete w.r.t. both data complexity and KB complexity [14].

As far as we know, the only practical approach for conjunctive query answering in $\mathcal{EL}$ has been proposed by Lutz et al. [11,12]. Our work can be regarded as an extension of their work by supporting role inclusions. A minor difference is that, in the ABox completion phase, we only generate one anonymous individual for each existential restriction, whereas their approach will generate one anonymous individual (called auxiliary object) for each subconcept of concepts used in $\mathcal{T}$. A major difference arises in the query rewriting phase, since our approach supports role inclusions. We firstly proposed a *base path criterion* and designed a base path storage schema to enable query rewriting based on this criterion. The query rewriting strategy in their approach is a special case of ours. Actually, as shown in the Section 4.5 for optimization, we can embed their query rewriting strategy into ours to handle the $\mathcal{ELH}$ part, while keeping the base path criterion dedicated for the "real" role inclusions in $\mathcal{EL}^+$.

Meanwhile, we can imagine a deterministic algorithm for query answering in $\mathcal{ELH}_{\mathcal{R}^+}$, which is a sub language of $\mathcal{SHIQ}$ [8]. Alternatively, DL-Lite [5] explores such a way that resorts simply to query rewriting without ABox completion. To employ either of them has to pay for an exponential blowup of the rewritten queries. Towards a practical implementation, query rewriting in our approach will only have a polynomial blowup, and we leverage RDB for query answering.

## 7 Conclusion

In this paper, we proposed a two-phase practical approach for query answering in $\mathcal{EL}^+$. In phase one, we do ABox completion using the notion of canonical individuals. For queries free of reverse trees, this phase is enough to provide sound and complete answers, w.r.t. arbitrary $\mathcal{EL}^+$ KBs. In phase two, we proposed the base path criterion and made the criterion applicable by the base path storage and query rewriting. A combination of the two phases guarantees sound and complete solutions to arbitrary queries on acyclic $\mathcal{EL}^+$ KBs.

As future work, we are planning to evolve the base path criterion, enabling queries on cyclic but regular $\mathcal{EL}^+$ ontologies. Development of more optimizations is also part of our ongoing work, especially reducing the blowup of base paths. Besides, for updating concerns, we will develop an incremental ABox completion, which is critical for the practical applications of our approach.

## Acknowledgements

The authors wish to thank Carsten Lutz for useful discussions during the writing of this paper.

## References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL Envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, pp. 364–369 (2005)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL Envelope Further. In: Proc. of the Workshop on OWL: Experiences and Directions (2008)
3. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In: Proc. of the Workshop on Description Logics (2006)
4. Baader, F., Nutt, W.: The Description Logic Handbook. In: Basic Description Logics, ch. 2. Cambridge University Press, Cambridge (2003)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable Description Logics for Ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence, pp. 602–607 (2005)
6. The Gene Ontology Consortium. Gene Ontology: Tool for the Unification of Biology. Journal of Nature Genetics 25, 25–29 (2000)
7. Dolin, R.H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F.M., Biron, P.V., Shabo, A.: HL7 Clinical Document Architecture, Release 2.0. Journal of American Medical Informatics Association 13(1), 30–39 (2006)
8. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive Query Answering for the Description Logic $\mathcal{SHIQ}$. Journal of Artificial Intelligence Research 31, 157–204 (2008)
9. IHTSDO. Systematized Nomenclature of Medicine C Clinical Terms, `http://www.ihtsdo.org/snomed-ct/`
10. Krotzsch, M., Rudolph, S., Hitzler, P.: Conjunctive Queries for a Tractable Fragment of OWL 1.1. In: Proc. of Int. Semantic Web Conf. (2007)
11. Lutz, C., Toman, D., Wolter, F.: Conjunctive Query Answering in $\mathcal{EL}$ using a Database System. In: Proc. of the Workshop on OWL: Experiences and Directions (2008)
12. Lutz, C., Toman, D., Wolter, F.: Conjunctive Query Answering in $\mathcal{EL}$ using a Database System. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (2009)
13. Rector, A., Horrocks, I.: Experience building a large, reusable Medical Ontology using a Description Logic with Transitivity and Concept Inclusions. In: Proc. of the Workshop on Ontological Engineering, AAAI Spring Symposium, Menlo Park (1997)
14. Rosati, R.: On Conjunctive Query Answering in $\mathcal{EL}$. In: Proc. of the Workshop on Description Logics (2007)