Data Science and Engineering
https://doi.org/10.1007/s41019-020-00150-0

SUMA: A Partial Materialization-Based Scalable Query Answering in OWL 2 DL

Xiaoyu Qin[1] · Xiaowang Zhang[1] · Muhammad Qasim Yasin[1] · Shujun Wang[1] · Zhiyong Feng[2] · Guohui Xiao[3]

Received: 30 May 2020 / Revised: 3 November 2020 / Accepted: 28 November 2020

Abstract
Ontology-mediated querying (OMQ) provides a paradigm for query answering according to which users not only query records at the database but also query implicit information inferred from ontology. A key challenge in OMQ is that the implicit information may be infinite, which cannot be stored at the database and queried by off -the -shelf query engine. The commonly adopted technique to deal with infinite entailments is query rewriting, which, however, comes at the cost of query rewriting at runtime. In this work, the partial materialization method is proposed to ensure that the extension is always finite. The partial materialization technology does not rewrite query but instead computes partial consequences entailed by ontology before the online query. Besides, a query analysis algorithm is designed to ensure the completeness of querying rooted and Boolean conjunctive queries over partial materialization. We also soundly and incompletely expand our method to support highly expressive ontology language, OWL 2 DL. Finally, we further optimize the materialization efficiency by role rewriting algorithm and implement our approach as a prototype system SUMA by integrating off-the-shelf efficient SPARQL query engine. The experiments show that SUMA is complete on each test ontology and each test query, which is the same as Pellet and outperforms PAGOdA. Besides, SUMA is highly scalable on large datasets.

Keywords Ontology reasoning · Materialization · Role rewriting

1 Introduction

With the rapid development of the semantic web, there are more and more applications based on ontology, especially, query answering applications [3]. Query answering can return implicit information that is not explicitly stored at the database but instead entailed by ontology to a user query. In this sense, it improves the quality of the answers compared to traditional database querying. The most popular query answering systems can be categorized into two major types: materialization-based and query rewriting-based.

The materialization approach precomputes all consequences entailed by ontologies (also known as the *universal model* or *chase*) offline so that the online query can be evaluated directly on the extended RDF data. Thus, it is preferred at online query performance-critical scenarios. PAGOdA [30] is based on materialization algorithm. It is scalable by delegating a large amount of the computational load to a datalog reasoner [21, 23] and using the hypertableau algorithm [22] only when necessary.

However, when the ontologies contain cyclic dependency relation, the universal model can be infinite [17]. The infinite extended data cannot be stored at the database and directly queried by the query engine. The infinite materialization is a significant challenge in materialization-based query answering systems. PAGOdA is incomplete in terms of infinite materialization (we proved it in Sect. 6).

The commonly adopted approach to deal with infinite materialization is query rewriting. Query rewriting techniques have been studied intensively and implemented in many systems, e.g., QuOnto [5], Mastro [7], Ontop [6] for

✉ Xiaowang Zhang
xiaowangzhang@tju.edu.cn

Xiaoyu Qin
xiaoyuqin@tju.edu.cn

[1] College of Intelligence and Computing, Tianjin University, Tianjin 300350, China

[2] College of Intelligence and Computing, Shenzhen Research Institute of Tianjin University, Tianjin University, Tianjin 300350, China

[3] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy

Published online: 19 January 2021

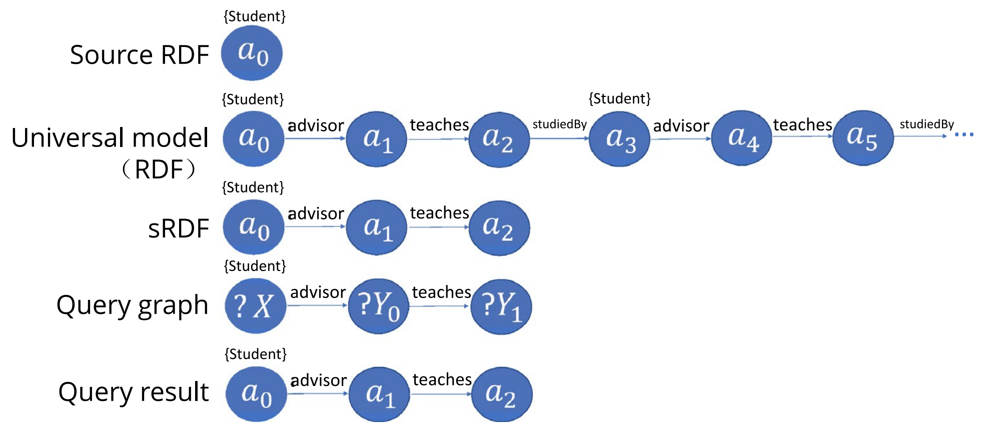


Fig. 1 A running example of partial materialization

DL-Lite, Grind [11] for $\mathcal{EL}$, Clipper [8] for Horn-$\mathcal{SHIQ}$. These systems do not materialize the data , but rather first rewrite the query according to the ontologies and mappings. Query rewriting uses a virtual RDF graph technique to avoid infinite materialization. However, it significantly increases the cost of query because rewriting is performed at runtime, and usually, it requires manual mapping. Moreover, the rewritten query can be exponentially large [26].

Besides query rewriting, the materialization-based ontology reasoner, e.g., Pellet [28], adopts a tableau algorithm with a roll-up technique [14] to solve infinite materialization. Pellet is not scalable for large datasets. It can only be applied to small and medium-sized datasets due to the high complexity of the tableau algorithm.

gOWL [19] proposes a partial materialization-based approach that deals with acyclic queries. The materialization algorithm of gOWL has a high time and space complexity due to its poor indexes for the storage and rules. Besides, gOWL cannot handle cyclic queries and Boolean queries, and its approximation rules lose most of the semantics of the OWL 2 DL.

There is also a hybrid approach [15] that computes the canonical model, which is always finite by reusing the anonymous individual name. And it rewrites the queries to remove wrong answers that the canonical model produces. But this rewriting algorithm cannot deal with role inclusion axioms. Then, a filter mechanism is proposed to replace the rewriting technique. The filter mechanism does not rewrite queries at runtime but filters spurious answers after query evaluation [17]. It supports role inclusions. However, it is still limited to lightweight ontology languages, DL-Lite$_\mathcal{R}$ [2].

Motivated by the users are mainly interested in the first few levels of the anonymous part of the universal models [10], in this paper, we describe a novel partial materialization approach. *Partial* means we do not compute all consequences entailed by ontology, but instead compute a subset of the universal model. *Partial* ensures the extended RDF data are always finite. Then, we propose a query analysis algorithm(QAA). QAA takes conjunctive queries as input, and its output indicates the size of the partial universal model. This algorithm is designed to ensure the partial materialization can always produce the same answers as a universal model for rooted conjunctive queries [16] and partial Boolean conjunctive queries in *DL-Lite*$^{\mathcal{N}}_{horn}$ [2]. Consider the following query Q : select $?X$ where $\{?X$ type Student. $?X$ advisor $?Y_0$. $?Y_0$ teaches $?Y_1$.$\}$. and the infinite universal model (the RDF), as shown in Fig. 1.

In this paper, we select only one part from the entire model (the sRDF) for answering the query Q as ans(sRDF, Q) = ans(RDF, Q). The sRDF is the partial universal model.

To make our approach unlimited to lightweight ontology languages, we soundly and incompletely generalize our approach to deal with OWL 2 DL by rewriting and approximation techniques. The approximation techniques apply to axioms whose semantics exceed *DL-Lite*$^{\mathcal{N}}_{horn}$. And, additional data structures are designed to preserve the semantics that approximation techniques may lose.

We implement our approach as a prototype system SUMA and integrate a role rewriting algorithm [27] to optimize materialization efficiency further. From a system perspective, SUMA allows us to design an offline modular architecture to integrate off-the-shelf efficient SPARQL [12] query engines. In this way, it makes online queries more efficient.

We validate our proposal in two cases: an evaluation in the finite universal model scenario and an evaluation in the infinite universal model scenario. In the former case, experiments are conducted on two widely adopted benchmark and two real datasets. In the latter case, we manually extend the LUBM [9] and UOBM [18] ontologies to evaluate query answering systems on an infinite universal model. These experiments confirm that: (i) the ontology reasoning algorithm used in PAGOdA cannot deal with infinite materialization, (ii) although Pellet is complete, it is not scalable and can only be used for small and medium data, (iii) SUMA is good at the trade-off the scalability and completeness. Experiments show that SUMA is highly efficient, only taking 124s to materialize LUBM(1000) and 411s to materialize UOBM(500). And, in each test query, it returns the same quality of answers as Pellet.

The rest of the paper is structured as follows. Section 2 introduces the basic notions. Section 3 presents the definition of QAA algorithm for rooted and Boolean conjunctive queries and gives a detailed proof. Section 4 shows how to approximate the OWL 2 DL axiom to *DL* axiom while preserving its semantics. Section 5 presents the architecture of SUMA and algorithms used at SUMA. The performance of SUMA on two benchmarks and real datasets is demonstrated in Sect. 6. Section 7 concludes this paper.

This work is an extension of the previous proceedings in [1]. In particular, (i) it extends QAA to support Boolean conjunctive queries that contain cyclic or fork structure, (ii) it adopts a role rewriting algorithm [27] to optimize materialization efficiency further, (iii) and it extends the experiments with role rewriting algorithm evaluation, gOWL system and YAGO dataset. This work confirms and extends the main finding from [1]: SUMA is good at the trade-off the scalability and completeness.

2 Preliminaries

In this section, we briefly introduce the syntax and semantics of description logics(DLs)$\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$, conjunctive query, and universal model.

2.1 Description Logics

Description Logic is a family of logics that have been studied and used in knowledge representation and reasoning. DLs underlie the standard Web Ontology Language OWL and OWL 2. In DLs, the elements of the domain are compiled into concepts (corresponding to unary predicates in first-order languages), and their properties are structured by means of roles (corresponding to binary predicates in first-order languages). Complex concepts and role expressions are made from atomic concepts and atomic role names. These names are connected by suitable constructors. The set of available constructors depends on the semantic of specific description logic. The richer the constructors that the description logic contains, the more complex the semantics that the description logic can capture.

Description logic knowledge base $\mathcal{K}$ consists of TBox ($\mathcal{T}$) and ABox ($\mathcal{A}$). A TBox typically consists of a set of axioms stating the inclusion between concepts and roles. The semantics of TBox is affected by the constructors. In an ABox, one can assert membership of objects (i.e., constants) in concepts, or that a pair of objects are connected by a role.

Let $\mathsf{N_I}$ be the individual set. In this paper, by default, we use a, b, c, d, e (with subscripts) to represent individual names. A, B, Z denote concept names, C (with subscripts) are concepts, P, S are role names, and R (with subscripts) are roles. Next, we present a brief overview of two different description logics, $\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$, and $\mathcal{SROIQ}$.

$\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$ defines roles and concepts as follows:

$$R\ := P \mid P^-, \qquad C\ := \bot \mid A \mid \geq m\, R.$$

$\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$ presents $\exists R$ as $\geq 1R$ and defines $(P^-)^-$ with P. Let $\mathsf{N_R^-}$ denote the set of roles.

A $\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$ $\mathcal{T}$ is a finite collection, including concept inclusions (CIs) axioms, that are in the form of $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq C$. A $\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$ ABox consists of concept assertions $A(a)$ and role assertions $P(a, b)$.

The semantics of $\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$ are defined by the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty domain. The function is denoted by $\cdot^{\mathcal{I}}$, which can map each A into the set $A^{\mathcal{I}}$, each P into the relation $P^{\mathcal{I}}$ and each a to an element $a^{\mathcal{I}}$. $A^{\mathcal{I}}$ and $P^{\mathcal{I}}$ are subsets of $\Delta^{\mathcal{I}}$ and $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, respectively. The $a^{\mathcal{I}}$ is an element of $\Delta^{\mathcal{I}}$. Besides, $\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$ implements the *unique name assumption* (UNA), that is, if v and w are distinct, then $v^{\mathcal{I}}$ is different from $w^{\mathcal{I}}$.

$\cdot^{\mathcal{I}}$ interprets each complex concept or role in the following ways: (1) $\bot^{\mathcal{I}} = \emptyset$ ($\bot$ is the bottom concept); (2) $(S^-)^{\mathcal{I}} = \{(v, w) \mid (w, v) \in S^{\mathcal{I}}\}$; and (3) $(\geq mS)^{\mathcal{I}} = \{w \mid \sharp\{v \mid (w, v) \in S^{\mathcal{I}}\} \geq m\}$. Here $\sharp$ denotes the cardinality.

The $\mathcal{I}$ satisfies a CIs axiom $\mathcal{T}_1$ in the form of $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq C$ if only if $\bigcap_{i=1}^{n} C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, denoted as $\mathcal{I} \vDash \mathcal{T}_1$. If $a^{\mathcal{I}} \in A^{\mathcal{I}}$ then $\mathcal{I} \vDash A(a)$ holds. If $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$ then $\mathcal{I} \vDash P(a, b)$ holds. If $\mathcal{I}$ satisfies all TBox and ABox axioms of $\mathcal{K}$ then $\mathcal{I}$ is a model of $\mathcal{K}$.

$\mathcal{SROIQ}$ is the underlying logic of OWL 2 DL. The concept of $\mathcal{SROIQ}$ is defined as: $C := \bot \mid \top \mid A \mid \neg A \mid \{a\} \mid \geq mR.A \mid \exists R.A$. Besides, $A \sqcup B$, $\forall R.A$, and $\leq nR.A$ can be rewritten to $\neg(\neg A \sqcap \neg B)$, $\neg\exists R.\neg A$, and $\neg \geq (n+1)R.A$, respectively. Enumeration $\{a_1, a_2, \ldots, a_n\}$ is equal to $\{a_1\} \sqcup \{a_2\} \sqcup \ldots \sqcup \{a_n\}$.

A $\mathcal{SROIQ}$ $\mathcal{K}$ consists of RBox $\mathcal{R}$, TBox $\mathcal{T}$ and ABox $\mathcal{A}$.

In addition to the concept inclusions (CIs) axioms contained in DL-Lite, a $\mathcal{SROIQ}$ $\mathcal{T}$ also includes disjointness axioms ($\mathrm{Dis}(C_1, C_2)$) and Equivalent concepts ($C_1 \equiv C_2$). Given an interpretation $\mathcal{I}$, we write $\mathcal{I} \vDash \mathrm{Dis}(C_1, C_2)$ if ${C_1}^{\mathcal{I}} \cap {C_2}^{\mathcal{I}} = \emptyset$, $\mathcal{I} \vDash \mathrm{Dis}(R_1, R_2)$ if ${R_1}^{\mathcal{I}} \cap {R_2}^{\mathcal{I}} = \emptyset$.

The RBox is a limited collection of either role inclusion axioms like $R_1 \sqsubseteq R_2$ or $R_1 \circ R_2 \sqsubseteq R_3$, or disjointness axioms in the form of $\mathrm{Dis}(R_1, R_2)$. The inverse role is denoted as $\mathrm{Inv}(R)$ with $\mathrm{Inv}(R) = R^-$, the symmetric role is denoted as $\mathrm{Sym}(R)$ (defined as $\mathrm{Inv}(R) \equiv R$), and the transitive role is denoted as $\mathrm{Trans}(R)$ (defined as $R \circ R \sqsubseteq R$). $\mathrm{Fun}(R)$ represents functional role. Given an interpretation

$\mathcal{I}$, we write $\mathcal{I} \vDash R_1 \sqsubseteq R_2$ if $R_1{}^{\mathcal{I}} \subseteq R_2{}^{\mathcal{I}}$, $\mathcal{I} \vDash R_1 \circ R_2 \sqsubseteq R_3$ if $R_1{}^{\mathcal{I}} \times R_2{}^{\mathcal{I}} \subseteq R_3{}^{\mathcal{I}}$.

A $\mathcal{SROIQ}$ ABox $\mathcal{A}$ without UNA includes individual equality $a \doteq b$ ($\doteq$ is called sameAs in OWL) and individual inequality $a \dot{\neq} b$. If $\mathcal{I} \vDash a \doteq b$ then $a^{\mathcal{I}} = b^{\mathcal{I}}$. If $\mathcal{I} \vDash a \dot{\neq} b$ then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. If the role R is functional, then $\mathcal{I} \vDash (\geq 2R \sqsubseteq \bot)$. Besides, if both $(a,b) \in R^{\mathcal{I}}$, $(a,d) \in R^{\mathcal{I}}$ and $b \dot{\neq} d \notin \mathcal{A}$, then $b \doteq d$.

2.2 Conjunctive Query

We use $\mathsf{N_V}$ to denote a collection of variables. $A(t)$ denotes the *concept* atomic form, and $P(t,t')$ denotes *role* atomic form with $t, t' \in \mathsf{N_I} \cup \mathsf{N_V}$. A *conjunctive query* (CQ) $q = \exists \mathbf{u} \psi(\mathbf{u}, \mathbf{v})$. It is making up of concept and role atoms. It connects these atoms by conjunction. The vector $\mathbf{v}$ consists of free variables. If $|\mathbf{v}| = 0$, we call q *Boolean*. The vector $\mathbf{u}$ comprises a collection of variables that are quantified. Since disconnected queries can be divided into connected subqueries for processing, this article only considers connected conjunctive queries. If a CQ is connected and not Boolean, it is a rooted CQ.

The notions of *answers* and *certain answers* of CQ are introduced as follows [15]. Let $q(\mathbf{v})$ be a CQ with $|\mathbf{v}| = k$, and $\mathcal{I}$ be an interpretation. The $\mathsf{N_T}$ is used for indicating the collection of all terms in q, that is $\mathsf{N_T} = \mathsf{N_I} \cup \mathsf{N_V}$. Let π be a *mapping* which maps each term of q to $\Delta^{\mathcal{I}}$ and each constant a to $a^{\mathcal{I}}$, we call $\mathcal{I}$ satisfies q under π if only if for every $A(t) \in q$, $\pi(t) \in A^{\mathcal{I}}$ and for every $P(t,t') \in q$, $(\pi(t), \pi(t')) \in P^{\mathcal{I}}$. The π is called a *match* for CQ in $\mathcal{I}$. The vector $\mathbf{a} = a_1 \dots a_k$ is an *answer* of q, when given a mapping π with $\pi(v_i) = a_i^{\mathcal{I}}$ $(i \leq k)$ and $\mathcal{I} \vDash^{\pi} q$. The ans$(q(\mathbf{v}), \mathcal{K})$ represents the collection of all answers of $q(\mathbf{v})$. Ind$(\mathcal{A})$ represents a set of individual names occurring in $\mathcal{A}$. Let's call $\mathbf{a}$ a certain answer when $\mathbf{a}$ is a subset of Ind$(\mathcal{A})$ and each model of $\mathcal{K}$ satisfies $q(\mathbf{a})$. The certain answer collection is denoted as cert$(q(\mathbf{v}), \mathcal{K})$.

2.3 Universal Model

Materialization is a forward chain algorithm that expands ABox according to the axioms in TBox. The ABox extension means expanding the ABox $\mathcal{A}$ to a universal model for the given KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. More specifically, during the materialization, the universal model is enriched by a set of additional individuals derived from existential and number restrictions axioms and additional assertions derived from CIs in $\mathcal{T}$.

A role R is called *generating in* $\mathcal{K}$ if there exists $a \in$ Ind$(\mathcal{A})$ and $R_1, \dots, R_n = R$ such that the followings hold: (agen) $\mathcal{K} \vDash \exists R_1(a)$ but $R_1(a,b) \notin \mathcal{A}$, for all $b \in$ Ind$(\mathcal{A})$ (written $a \rightsquigarrow c_{R_1}$); (rgen) for $i < n$, $\mathcal{T} \vDash \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$ (written $c_{R_i} \rightsquigarrow c_{R_{i+1}}$). If R is generating in $\mathcal{K}$, then c_R is called an *anonymous* individual. And, the anonymous individual collection is denoted as $\mathsf{N}_{\mathsf{I}}^{\mathcal{T}}$, which is disjoint from Ind$(\mathcal{A})$. A new assertion $C_2(a)$ will be included in universal model, if the ABox $\mathcal{A}$ contains $C_1(a)$, TBox $\mathcal{T}$ contains $C_1 \sqsubseteq C_2$ and ABox $\mathcal{A}$ does not contain $C_2(a)$.

The *canonical interpretation* $\mathcal{I}_{\mathcal{K}}$ for $\mathcal{K}$ is defined as follows:

- $\Delta^{\mathcal{I}_{\mathcal{K}}} = \text{Ind}(\mathcal{A}) \cup \{c_R \mid R \in N_R^-, R \text{ is generating in } \mathcal{K}\}$;
- $a^{\mathcal{I}_{\mathcal{K}}} = a$, for all $a \in \text{Ind}(\mathcal{A})$;
- $A^{\mathcal{I}_{\mathcal{K}}} = \{a \in \text{Ind}(\mathcal{A}) \mid \mathcal{K} \vDash A(a)\} \cup \{c_R \in \Delta^{\mathcal{I}_{\mathcal{K}}} \mid \mathcal{T} \vDash \exists R^- \sqsubseteq A\}$;
- $P^{\mathcal{I}_{\mathcal{K}}} = \{(a,b) \in \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) \mid P(a,b) \in \mathcal{A}\}$
  $\cup \{(d, c_P) \in \Delta^{\mathcal{I}_{\mathcal{K}}} \times \mathsf{N}_{\mathsf{I}}^{\mathcal{T}} \mid d \rightsquigarrow c_P\}$
  $\cup \{(c_{P^-}, d) \in \mathsf{N}_{\mathsf{I}}^{\mathcal{T}} \times \Delta^{\mathcal{I}_{\mathcal{K}}} \mid d \rightsquigarrow c_{P^-}\}$.

A path in $\mathcal{I}_{\mathcal{K}}$ is a finite sequence $ac_{R_1} \cdots c_{R_n}$ $(n \geq 0)$, such that $a \in$ Ind$(\mathcal{A})$ and $R_1, \dots, R_n$ satisfy (**agen**) and (**rgen**) (that is, $a \rightsquigarrow c_{R_1}$ and $c_{R_i} \rightsquigarrow c_{R_{i+1}}$, for $1 \leq i < n$). The last element of σ in a path is denoted by tail(σ).

The *universal model* $\mathcal{U}_{\mathcal{K}}$ is defined as follows:

- $\Delta^{\mathcal{U}_{\mathcal{K}}} = \{a \cdot c_{R_1} \cdots c_{R_n} \mid a \in \text{Ind}(\mathcal{A}),\quad n \geq 0,\ a \rightsquigarrow c_{R_1} \rightsquigarrow \cdots \rightsquigarrow c_{R_n}\}$,
- $a^{\mathcal{U}_{\mathcal{K}}} = a$, for all $a \in \text{Ind}(\mathcal{A})$;
- $A^{\mathcal{U}_{\mathcal{K}}} = \{\sigma \in \Delta^{\mathcal{U}_{\mathcal{K}}} \mid \text{tail}(\sigma) \in A^{\mathcal{I}_{\mathcal{K}}}\}$;
- $P^{\mathcal{U}_{\mathcal{K}}} = \{(a,b) \in \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) \mid P(a,b) \in \mathcal{A}\} \cup \{(\sigma, \sigma \cdot c_P) \in \Delta^{\mathcal{U}_{\mathcal{K}}} \times \Delta^{\mathcal{U}_{\mathcal{K}}} \mid \text{tail}(\sigma) \rightsquigarrow c_P\} \cup \{(\sigma \cdot c_{P^-}, \sigma) \in \Delta^{\mathcal{U}_{\mathcal{K}}} \times \Delta^{\mathcal{U}_{\mathcal{K}}} \mid \text{tail}(\sigma) \rightsquigarrow c_{P^-}\}$.

The difference between a canonical interpretation and a universal model is that the canonical interpretation is always finite. It ensures that the extended ABox is finite by reusing the symbols of anonymous individuals. However, this reuse mechanism can lead to the canonical model producing wrong answers to conjunctive queries under the certain answer semantic. We compare the definitions of *canonical interpretation* $\mathcal{I}_{\mathcal{K}}$ and *universal model* $\mathcal{U}_{\mathcal{K}}$ by Examples 1 and 2.

Example 1 Let $\mathcal{K}$ consist of $\mathcal{T} = \{B \sqsubseteq \exists S, \exists S^- \sqsubseteq \exists S\}$ and $\mathcal{A} = \{B(d_0)\}$.

Then $\Delta_{\mathcal{K}}^{\mathcal{I}} = \{d_0, d_1\}$, $B^{\mathcal{I}_{\mathcal{K}}} = \{d_0\}$, and $S^{\mathcal{I}_{\mathcal{K}}} = \{(d_0, d_1), (d_1, d_1)\}$.

$\Delta_{\mathcal{K}}^{\mathcal{U}} = \{d_0, d_1, d_2, d_3, \dots\}$, $B^{\mathcal{U}_{\mathcal{K}}} = \{d_0\}$, and $S^{\mathcal{U}_{\mathcal{K}}} = \{(d_0, d_1), (d_1, d_2), \dots\}$.

Let q be $\exists v\ S(v,v)$. If $\pi(v) = d_1$, then $\mathcal{I}_{\mathcal{K}} \vDash^{\pi} q$, however, $\mathcal{U}_{\mathcal{K}} \nvDash^{\pi} q$. Thus, $\mathcal{K} \nvDash^{\pi} q$, the canonical interpretation $\mathcal{I}_{\mathcal{K}}$ produces wrong answers.

Example 2 Let $\mathcal{K}$ consist of $\mathcal{T} = \{\text{Student} \sqsubseteq \exists\text{advisor}, \exists\text{advisor}^- \sqsubseteq \exists\text{teaches}, \exists\text{teaches}^- \sqsubseteq \exists\text{studiedBy}, \exists\text{studiedBy}^- \sqsubseteq \text{Student}\}$, and $\mathcal{A} = \{\text{Student}(a_0)\}$.

The universal model of this example is shown in Fig. 1.

Kontchakov et al. proposed that for every consistent $DL\text{-}Lite^{\mathcal{N}}_{horn}$ KB $\mathcal{K}$ and every CQ q, we have $\text{cert}(q, \mathcal{K}) = \text{ans}(q, \mathcal{U}_{\mathcal{K}})$ [15].

3 *n*-step Universal Model *DL-Lite*$^{\mathcal{N}}_{horn}$

In this section, we present the definition of n-step universal model and query analysis algorithm. The n-step universal model is a implementation of the partial materialization idea. And the query analysis algorithm is proposed to ensure the n-step universal model always has the same answers as the universal model.

3.1 *n*-step Universal Model

Query answering over $\mathcal{A}$ with $\mathcal{T}$ can be reduced to query answering over the universal model, as shown in preliminaries. However, the universal model can be infinite.

Considering such axioms, it satisfies three characteristics. Firstly, it belongs to the concept of inclusions. Secondly, its head and body directly or indirectly contain both existential quantifiers. Thirdly, the roles included in this axiom are inverse to each other. We refer to this kind of axiom as cyclic existential quantifiers axioms (CEQ, for short) in this paper. The simple form of CEQ axioms is $\exists R^- \sqsubseteq \exists R$ or $\exists R^- \sqsubseteq A$, $A \sqsubseteq \exists R$. When ontology contains CEQ axioms, the universal model is infinite [17], as shown in Examples 1 and 2. The infinite RDF data cannot be directly stored and queried with off the shelf query engine.

We propose an n-step universal model to replace the possible infinite universal model. Intuitively, the process of materialization is to extend ABox to a universal model. The process can be thought of as a sequence $U = \{\text{ABox}, \mathcal{U}^1_{\mathcal{K}}, \mathcal{U}^2_{\mathcal{K}}, \ldots, \mathcal{U}^n_{\mathcal{K}}, \ldots\}$. And, $\mathcal{U}^i_{\mathcal{K}} \subseteq \mathcal{U}^{i+1}_{\mathcal{K}}, i < |U|$. The (agen) and (rgen) (see Sect. 2) are the fundamental reasons for expanding $\mathcal{U}^i_{\mathcal{K}}$ to $\mathcal{U}^{i+1}_{\mathcal{K}}$. The element $\mathcal{U}^n_{\mathcal{K}}$ of U is called the n-step universal model in our method. $\mathcal{U}^n_{\mathcal{K}}$ is always finite. The difference between partial materialization and materialization is that the core of partial materialization is to calculate the n-step universal model, while materialization computes the whole model. We ensure our materialization is always finite by selecting $\mathcal{U}^n_{\mathcal{K}}$ from the $\mathcal{U}_{\mathcal{K}}$ for query answering.

To formalize $\mathcal{U}^n_{\mathcal{K}}$, we require some preliminary definitions. We label R as *n-step generating* in $\mathcal{K}$, when a is an individual of $\text{Ind}(\mathcal{A})$, and the $R = R_1 \ldots R_n$ satisfies (agen) and (rgen). The $\mathsf{N}^{\mathsf{n}}_{\mathsf{R}}$ denotes the set of roles that are $\leq n$-step generating in $\mathcal{K}$. The n-step canonical interpretation is defined as follows:

Definition 1 n-step canonical interpretation

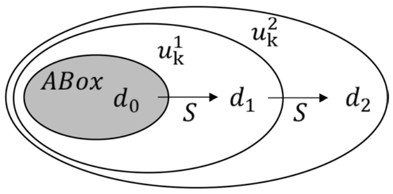


Fig. 2 The 2-step universal model of Example 3

$$\begin{aligned}
&\Delta^{\mathcal{I}^n_{\mathcal{K}}} = \text{Ind}(\mathcal{A}) \cup \{c_R | R \in \mathsf{N}^{\mathsf{n}}_{\mathsf{R}}\},\\
&a^{\mathcal{I}^n_{\mathcal{K}}} = a, \text{for all } a \in \text{Ind}(\mathcal{A}),\\
&A^{\mathcal{I}^n_{\mathcal{K}}} = \{a \in \text{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{c_R \in \Delta^{\mathcal{I}^n_{\mathcal{K}}} \mid \mathcal{T} \models \exists R^- \sqsubseteq \mathcal{A}\},\\
&P^{\mathcal{I}^n_{\mathcal{K}}} = \{(a,b) \in \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) | P(a,b) \in \mathcal{A}\} \cup \{(d, c_p)\\
&\in \Delta^{\mathcal{I}^n_{\mathcal{K}}} \times N^{\mathcal{T}}_I | d \rightsquigarrow c_p\} \cup \{(c_{P^-}, d) \in N^{\mathcal{T}}_I \times \Delta^{\mathcal{I}^n_{\mathcal{K}}} | d \rightsquigarrow c_{P^-}\}.
\end{aligned}$$

All roles included in the n-step canonical interpretation must be $\leq n$-step generating in $\mathcal{K}$. All anonymous variables are witness of roles which are $\leq n$-step generating in $\mathcal{K}$. Based on the definition of the n-step canonical interpretation, we can derive the definition of n-step universal model ($\mathcal{U}^n_{\mathcal{K}}$).

Definition 2 n-step universal model

$$\begin{aligned}
&\Delta^{\mathcal{U}^n_{\mathcal{K}}} = \{a \cdot c_{R_1} \cdots c_{R_l} \mid a \in \text{Ind}(\mathcal{A}), R_l \in \mathsf{N}^{\mathsf{n}}_{\mathsf{R}}, a \rightsquigarrow c_{R_1} \rightsquigarrow\\
&\ldots \rightsquigarrow c_{R_l}\},\\
&a^{\mathcal{U}^n_{\mathcal{K}}} = a, \text{ for all } a \in \text{Ind}(\mathcal{A}),\\
&A^{\mathcal{U}^n_{\mathcal{K}}} = \{\sigma \in \Delta^{\mathcal{U}^n_{\mathcal{K}}} \mid tail(\sigma) \in A^{\mathcal{I}^n_{\mathcal{K}}}\},\\
&P^{\mathcal{U}^n_{\mathcal{K}}} = \{(a,b) \in \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) \mid P(a,b) \in \mathcal{A}\} \cup \{(\sigma, \sigma\\
&\cdot c_P) \in \Delta^{\mathcal{U}^n_{\mathcal{K}}} \times\\
&\Delta^{\mathcal{U}^n_{\mathcal{K}}} \mid tail(\sigma) \rightsquigarrow c_P\} \cup \{(\sigma \cdot c_{P^-}, \sigma) \in \Delta^{\mathcal{U}^n_{\mathcal{K}}} \times \Delta^{\mathcal{U}^n_{\mathcal{K}}} \mid tail(\sigma)\\
&\rightsquigarrow c_{P^-}\}.
\end{aligned}$$

Example 3 We use Example 1 to illustrate $\mathcal{U}^n_{\mathcal{K}}$. $\Delta^{\mathcal{U}^n_{\mathcal{K}}} = \{d_0, d_1, \ldots, d_n\}, {d_0}^{\mathcal{U}^n_{\mathcal{K}}} = d_0$, $B^{\mathcal{U}^n_{\mathcal{K}}} = \{d_0\}$, $S^{\mathcal{U}^n_{\mathcal{K}}} = \{(d_0, d_1), (d_1, d_2), \ldots, (d_{n-1}, d_n)\}$. The graph of the 2-step universal model is shown in Fig. 2.

Example 4 The 2-step universal model (sRDF) of Example 2 is shown in Fig. 1.

3.2 QAA Based on Rooted Conjunctive Queries

We design a query analysis algorithm (QAA) to ensure that the n-step universal model can always compute the same answer as the universal model. QAA takes a rooted query as input and calculates the number of quantified variables n. Obviously, n can be calculated in $\mathcal{O}(|\mathsf{N}_{\mathsf{T}}|)$. The number of quantified variables denotes the step of the universal model. If the step size is n, then the n-step universal model, i.e., the $\mathcal{U}^n_{\mathcal{K}}$ in U, can produce the same answers for q as the universal model. This method is proved in Theorem 2. We define a new triple relation on q. σ denotes a path that consists of terms. δ represents a path that includes roles.

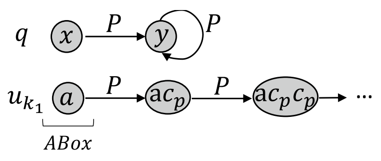


Fig. 3 The universal model of $\mathcal{K}_1$

Definition 3 Let $q = \exists \mathbf{u}\varphi(\mathbf{u},\mathbf{v})$ be a CQ, $\mathcal{T}$ be a TBox and π be a mapping, we define a triple relation $f_\rho = \cup_{i\geq 0} f_\rho^i \subseteq \mathsf{N_T} \times \mathsf{N_T}^* \times \mathsf{N_R^-}^*$ with $\rho = \{t \mid t \in \mathsf{N_T}, \pi(t) \in \mathrm{Ind}(\mathcal{A})\}$, where

$$f_\rho^0 = \{(t,t,\varepsilon) \mid t \in \rho\};$$
$$f_\rho^{i+1} = f_\rho^i \cup \{(t,\sigma st,\delta SR) \mid (s,\sigma s,\delta S) \in f_\rho^i, R(s,t) \in q, tail(\pi(s)) \rightsquigarrow tail(\pi(t))\} \cup \{(t,\sigma,\delta) \mid (s,\sigma s,\delta R^-) \in f_\rho^i, R(s,t) \in q, tail(\pi(s)) \rightsquigarrow tail(\pi(t))\}.$$

A path $\sigma = t_0 \cdot t_1 \cdots t_{n-1} \cdot t_n$ is a *certain path* of q, if (t_0) is mapped to $\mathrm{Ind}(\mathcal{A})$ and all other terms are mapped to $\mathsf{N}_\mathrm{I}^{\mathcal{T}}$. A certain path collection is represented as $\mathrm{CertPath}(q,\pi)$. The *max certain path* is defined as $\mathrm{MaxCertPath}(q,\pi) := \{\sigma \mid |\sigma_i| \leq |\sigma|$, for all $\sigma_i \in \mathrm{CertPath}(q,\pi)\}$. Let π be a mapping. We set the *depth* of q as $\mathrm{dep}(q,\pi) := |\sigma| - 1$, with $\sigma \in \mathrm{MaxCertPath}(q,\pi)$.

The anonymous part of the universal model is a forest-shaped structure, as shown in Examples 3 and 4. Thus, the subquery that is matched to the anonymous part of the universal model must be a forest-shaped structure. If a term t in q wants to be mapped to an anonymous part of $\mathcal{U}_\mathcal{K}$, it can only be mapped in this way, $\pi(t) = \pi(t_0) \cdot c_{R_1} \cdots c_{R_n}$ with $(t,\sigma,\delta) \in f_\rho$, where $\sigma = t_0 t_1 \cdots t_n$ is a certain path, and $\delta = R_1 R_2 \cdots R_n$.

Example 5 Let $q(x)$ be a cyclic CQ of the following form: $\exists\, y P(x,y) \wedge P(y,y)$, $\mathcal{T} = \{A \sqsubseteq \exists P, \exists P^- \sqsubseteq \exists P\}$.

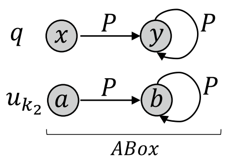
Given the first knowledge base $\mathcal{K}_1 = \{\mathcal{T},\mathcal{A}_1\}$ and $\mathcal{A}_1 = \{A(a)\}$. As shown in Fig. 3, the universal model is heterogeneous to the query q. Thus, $\mathrm{ans}(q,\mathcal{U}_\mathcal{K}) = \mathrm{cert}(q,\mathcal{K}_1) = \emptyset$.

Given the second knowledge base $\mathcal{K}_2 = \{\mathcal{T},\mathcal{A}_2\}$ and $\mathcal{A}_2 = \{A(a), P(a,b), P(b,b)\}$. As shown Fig. 4, the universal model is homogeneous to the query q. $\pi = \{x \rightarrow a, y \rightarrow b\}$. $f_\rho = f_\rho^0 = \{(x,x,\varepsilon),(y,y,\varepsilon)\}$. Thus, $\mathrm{dep}(q,\pi) = 0$. $\mathrm{ans}(q,\mathcal{U}_\mathcal{K}) = \mathrm{cert}(q,\mathcal{K}_1) = \{a\}$.

These two examples demonstrate that the cyclic part of the query only can be mapped into the ABox part of the universal model because the anonymous part of the universal model is a forest-shaped structure.

Example 6 Let $q(x)$ be a fork-shaped CQ of the following form: $\exists\, y P(x,y) \wedge P(z,y)$, $\mathcal{T} = \{A \sqsubseteq \exists P, \exists P^- \sqsubseteq \exists P\}$.

Fig. 4 The universal model of $\mathcal{K}_2$



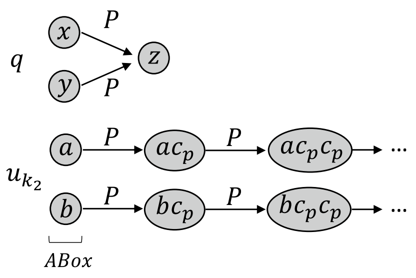


Fig. 5 The universal model of $\mathcal{K}_1$

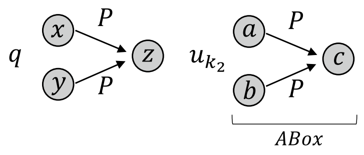


Fig. 6 The universal model of $\mathcal{K}_2$

Consider $\mathcal{K}_1 = \{\mathcal{T},\mathcal{A}_1\}$ and $\mathcal{A}_1 = \{A(a), A(b)\}$. Then, as shown in Fig. 5, the universal model is heterogeneous to the query q. Thus, $\mathrm{ans}(q,\mathcal{U}_\mathcal{K}) = \mathrm{cert}(q,\mathcal{K}_1) = \emptyset$.

Let $\mathcal{K}_2 = \{\mathcal{T},\mathcal{A}_2\}$, where $\mathcal{A}_1 = \{A(a), A(b), P(a,c), P(b,c)\}$. Then, as shown in Fig. 6, the universal model is homogeneous to the query q. $\pi = \{x \rightarrow a, z \rightarrow b, y \rightarrow c\}$. $f_\rho = f_\rho^0 = \{(x,x,\varepsilon),(y,y,\varepsilon),(z,z,\varepsilon)\}$. Thus, $\mathrm{dep}(q,\pi) = 0$. $\mathrm{ans}(q,\mathcal{U}_\mathcal{K}) = \mathrm{cert}(q,\mathcal{K}_1) = \{a,b\}$.

These two examples confirm that not only a cyclic query, but also a fork query, the fork part of the query can only be mapped into the ABox part of the universal model, due to the anonymous part of the universal model is a forest-shaped structure.

Theorem 1 *For every consistent DL-Lite$_{horn}^{\mathcal{N}}$ KB $\mathcal{K}$, every rooted conjunctive query $q = \exists \mathbf{u}\varphi(\mathbf{u},\mathbf{v})$, and every mapping π, with $\mathrm{dep}(q,\pi) = n$, we have $\mathcal{U}_\mathcal{K} \models^\pi q$ if only if $\mathcal{U}_\mathcal{K}^n \models^\pi q$.*

Proof (⇒) For every π with $\mathrm{dep}(q,\pi) = n$, then there exists a max certain path $\sigma = t_0 t_1 \cdots t_{n-1} t_n$ and $R(t_{n-1},t_n) \in q$. Thus, R is n-step generating, and all other roles are $\leq n$-step generating. By definition of $\mathcal{U}_\mathcal{K}^n$, we have that for every $R \in q$, if

$(a,b) \in R^{\mathcal{U}_\mathcal{K}}$ then $a \in \Delta^{\mathcal{U}^n_\mathcal{K}}$ and $b \in \Delta^{\mathcal{U}^n_\mathcal{K}}$. Thus, $(a,b) \in R^{\mathcal{U}^n_\mathcal{K}}$. Since q is connected, for every $A \in q$, suppose $a \in A^{\mathcal{U}_\mathcal{K}}$, then $(a,*) \in R^{\mathcal{U}_\mathcal{K}}$ or $(*,a) \in R^{\mathcal{U}_\mathcal{K}}$. Thus, $a \in \Delta^{\mathcal{U}^n_\mathcal{K}}$ and $a \in A^{\mathcal{U}^n_\mathcal{K}}$. In conclusion, $\mathcal{U}^n_\mathcal{K} \models q$.

($\Leftarrow$) For every $R \in q$, if $(a,b) \in R^{\mathcal{U}^n_\mathcal{K}}$ then $a \in \Delta^{\mathcal{U}^n_\mathcal{K}}$ and $b \in \Delta^{\mathcal{U}^n_\mathcal{K}}$. Because $\Delta^{\mathcal{U}^n_\mathcal{K}}$ is a subset of $\Delta^{\mathcal{U}}_\mathcal{K}$, $a \in \Delta^{\mathcal{U}}_\mathcal{K}$, $b \in \Delta^{\mathcal{U}}_\mathcal{K}$ and $(a,b) \in R^{\mathcal{U}_\mathcal{K}}$. For every $A \in q$, if $a \in A^{\mathcal{U}^n_\mathcal{K}}$, then $a \in \Delta^{\mathcal{U}^n_\mathcal{K}}$. Thus $a \in \Delta^{\mathcal{U}}_\mathcal{K}$ and $a \in A^{\mathcal{U}_\mathcal{K}}$.

Therefore, $\mathcal{U}^n_\mathcal{K} \models q(\mathbf{a},\mathbf{b})$ if only if $\mathcal{U}_\mathcal{K} \models q(\mathbf{a},\mathbf{b})$. □

Let f_ρ is a function if for every term t, $f_\rho(t)$ is singleton set or if $(t,\sigma,\delta) \in f(t)$ and $(t,\sigma',\delta') \in f(t)$, then $\delta = \delta'$.

Lemma 1 *For every mapping, if $\mathcal{U}_\mathcal{K} \models^\pi q$, then f_ρ is a function and every $\sigma \in$ MaxCertPath(q,π) is finite and* dep$(q,\pi) \le |\mathbf{u}|$.

Proof Suppose f_ρ is not a function, then there exists $t \in \mathsf{N_T}$, with $(t,\sigma_i,\delta_i) \in f_\rho(t)$ and $(t,\sigma_j,\delta_j) \in f_\rho(t)$, where $\delta_i \neq \delta_j$. We labeled δ_i and δ_j as $\delta_i = R^0_i \cdot R^1_i \cdots R^{n_i}_i$ and $\delta_j = R^0_j \cdot R^1_j \cdots R^{n_j}_j$, respectively. Thus, $\mathcal{U}_\mathcal{K} \not\models^\pi q$ due to $c_{R^0_i} \cdots c_{R^{n_i}_i} \neq c_{R^0_j} \cdots c_{R^{n_j}_j}$. This creates a contradiction.

Because $\mathsf{N_T}$ is finite, if σ is not finite, then there exists t', with $\sigma = \sigma_1 t' \sigma_2 t' \sigma_3$. Thus, $(t', \sigma_1 t', \delta_1) \in f_\rho(t')$ and $(t', \sigma_1 t' \sigma_2 t', \delta_2) \in f_\rho(t')$. By the definition of f_ρ, we have that δ_1 is a subsequence of δ_2 because $\sigma_1 t'$ is a subsequence of $\sigma_1 t' \sigma_2 t'$. Thus, $f_\rho(t)$ is not a function. However, we have proved that if $\mathcal{U}_\mathcal{K} \models^\pi q$, then f_ρ is a function. This creates a contradiction.

Suppose dep$(q,\pi) > |\mathbf{u}|$, then there exists a σ with $|\sigma| > |\mathbf{u}| + 1$. Because free variables can only be mapped into Ind$(\mathcal{A})$, a quantified variable repeatedly appears in the path σ exists. Thus, σ is infinite. We have proved that if $\mathcal{U}_\mathcal{K} \models^\pi q$, then every $\sigma \in$ MaxCertPath(q,π) is finite. This creates a contradiction. □

Based on Lemma 1 and Theorem 1, we can conclude that, for every rooted query, we extend the model at most $|\mathbf{u}|$ steps. The core of the QAA:

Theorem 2 *For each consistent DL-Lite$^\mathcal{N}_{horn}$ KB $\mathcal{K}$ and each rooted conjunctive query $q = \exists \mathbf{u} \varphi(\mathbf{u},\mathbf{v})$, with $|\mathbf{u}| = n$, we have* cert$(q,\mathcal{K})$ = ans$(q,\mathcal{U}^n_\mathcal{K})$.

Proof Kontchakov et al. proposed that cert$(q,\mathcal{K})$ = ans$(q,\mathcal{U}_\mathcal{K})$ (see Sect. 2). Thus, we only need to proof ans$(q,\mathcal{U}_\mathcal{K})$ = ans$(q,\mathcal{U}^n_\mathcal{K})$.

($\Rightarrow$) Lemma 1 shows that for every mapping π, if $\mathcal{U}_\mathcal{K} \models^\pi q$, then $n^* =$ dep$(q,\pi) \le |\mathbf{u}|$. Based on Theorem 1, we can conclude that $\mathcal{U}^{n^*}_\mathcal{K} \models^\pi q$. Because $n^* \le |\mathbf{u}|$, $\Delta^{\mathcal{U}^{n^*}_\mathcal{K}} \subseteq \Delta^{\mathcal{U}^n_\mathcal{K}} \subseteq \Delta^{\mathcal{U}}_\mathcal{K}$. We can conclude that $\mathcal{U}^n_\mathcal{K} \models^\pi q$.

($\Leftarrow$) Because $\Delta^{\mathcal{U}^n_\mathcal{K}} \subseteq \Delta^{\mathcal{U}}_\mathcal{K}$, if $\mathcal{U}^n_\mathcal{K} \models^\pi q$ then $\mathcal{U}_\mathcal{K} \models^\pi q$.

In conclusion, ans$(q,\mathcal{U}_\mathcal{K})$ = ans$(q,\mathcal{U}^n_\mathcal{K})$, that is, cert$(q,\mathcal{K})$ = ans$(q,\mathcal{U}_\mathcal{K})$. □

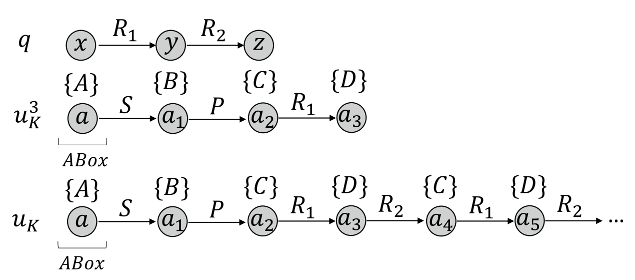


Fig. 7 The universal model of $\mathcal{K}$

Example 7 The example in Sect. 1 proves our idea. Under the $\pi = \{x \to a_0, y_0 \to a_1, y_1 \to a_2\}$, we can conclude that $f_\rho = \{(x,x,\varepsilon), (y_0, xy_0, \text{advisor})(y_1, xy_0y_1, \text{advisor} \cdot \text{teaches})\}$ CertPath$(q,\pi) = \{\sigma_1,\sigma_2,\sigma_3\}$ with $\sigma_1 = x$, $\sigma_2 = xy_0$, $\sigma_3 = xy_0y_1$. Then MaxCertPath$(q,\pi) = \{\sigma_3\}$, and, dep$(q,\pi) = 2$. Obviously, the dep(q,π) is not greater than the number of quantified variables, thus, the sRDF(2- step universal model) makes ans(sRDF, Q) = ans(RDF, Q).

3.3 QAA Based on Boolean Conjunctive Queries

A Boolean conjunctive query is in the form of $q = \exists \mathbf{u} \psi(\mathbf{u})$. It is a special case of conjunctive queries $q = \exists \mathbf{u} \psi(\mathbf{u},\mathbf{v})$, with $|\mathbf{v}| = 0$. As mentioned in Sect. 2, all non-connected queries can be decomposed into connected subqueries for processing, so we only consider connected Boolean queries. Because Boolean conjunctive query does not have a variable that must be mapped into ABox, the query analysis algorithm does not explicitly apply to Boolean conjunctive queries. As shown in Example 8, the query analysis algorithm is unsound for Boolean conjunctive queries $q(x)$, that is, when $\mathcal{U}^n_\mathcal{K}$ does not satisfy q, it is possible for $\mathcal{K}$ to satisfy q.

Example 8 Let $q(x)$ be a Boolean CQ of the following form: $\exists\, x\, y\, z\, (R_1(x,y) \wedge R_2(y,z))$, $\mathcal{T} = \{A \sqsubseteq \exists S, \exists S^- \sqsubseteq B, B \sqsubseteq \exists P, \exists P^- \sqsubseteq C, C \sqsubseteq \exists R_1, \exists R_1{}^- \sqsubseteq D, D \sqsubseteq \exists R_2, \exists R_2{}^- \sqsubseteq C\}$ and $\mathcal{A} = \{A(a)\}$. Figure 7 shows the graph of 3-step universal model and query. Obviously, the 3-step universal model does not satisfy q, but the universal model does.

To extend QAA to support Boolean conjunctive queries, we firstly summarize the general properties of the universal model.

If the ontology contains the CEQ axioms, the universal model is infinite.

The anonymous part of the universal model is forest-like, and is heterogeneous to the cyclic and fork structures.

The two properties are explained or proved in the preceding two sections. Based on the two attributes, we can conclude that if the Boolean conjunctive query contains a cyclic structure or fork structure and the Boolean conjunctive query can be satisfied, then there must be a variable mapped to ABox. Therefore, the Boolean conjunctive queries which contain a cyclic or fork structure can be directly solved by QAA. To verify this, we first extend the triple relation f_ρ defined in the rooted queries to Boolean queries by modifying the definition of ρ.

Definition 4 Let $q = \exists \mathbf{u}\varphi(\mathbf{u}, \mathbf{v})$ be a CQ, $\mathcal{T}$ be a TBox and π be a mapping, we define a triple relation $f_\rho = \cup_{i\geq 0} f_\rho^i \subseteq \mathsf{N}_\mathsf{T} \times \mathsf{N}_\mathsf{T}^* \times \mathsf{N}_\mathsf{R}^{-*}$ with $\rho = \{t \mid t \in \mathsf{N}_\mathsf{T}, \pi(t) \in \mathrm{Ind}(\mathcal{A})$ or there is no t'such that $tail(\pi(t')) \rightsquigarrow tail(\pi(t))\}$ where

$$f_\rho^0 = \{(t, t, \varepsilon) \mid t \in \rho\};$$
$$f_\rho^{i+1} = f_\rho^i \cup \{(t, \sigma st, \delta SR) \mid (s, \sigma s, \delta S) \in f_\rho^i, R(s,t) \in q, tail(\pi(s)) \rightsquigarrow tail(\pi(t))\} \cup \{(t, \sigma, \delta) \mid (s, \sigma s, \delta R^-) \in f_\rho^i, R(s,t) \in q, tail(\pi(s)) \rightsquigarrow tail(\pi(t))\}.$$

Lemma 2 *For each consistent DL-Lite$_{horn}^{\mathcal{N}}$* KB $\mathcal{K}$ *and each Boolean conjunctive query $q = \exists \mathbf{u}\varphi(\mathbf{u})$ which contains a cyclic structure or fork structure, we have that if $\mathcal{U}_\mathcal{K}$ satisfies q and q contains a cyclic structure or fork structure, then there are variables in the query that must be mapped to ABox.*

Proof Assuming that all variables in q are mapped to anonymous variables, if $\mathcal{U}_\mathcal{K}$ satisfies q and q contains a cyclic structure or fork structure, then there must be a variable $t \in \mathsf{N}_\mathsf{T}$, such that $(t, \sigma_i, \delta_i) \in f_\rho(t)$ and $(t, \sigma_j, \delta_j) \in f_\rho(t)$, where $head(\sigma_i) \cdot \delta_i \neq head(\sigma_j) \cdot \delta_j$. The δ_i and δ_j are labeled as $\delta_i = R_i^0 \cdot R_i^1 \cdots R_i^{n_i}$ and $\delta_j = R_j^0 \cdot R_j^1 \cdots R_j^{n_j}$, respectively. Thus, $\mathcal{U}_\mathcal{K} \nvDash^\pi q$ due to $\pi(head(\sigma_i)) \cdot c_{R_i^0} \cdots c_{R_i^{n_i}} \neq \pi(head(\sigma_j)) \cdot c_{R_j^0} \cdots c_{R_j^{n_j}}$. This creates a contradiction. Thus, if $\mathcal{U}_\mathcal{K}$ satisfies Boolean conjunctive query q and q contains a cyclic structure or fork structure, then there are variables in the query that must be mapped to ABox. □

Example 9 Let $q(x)$ be a fork-shaped CQ of the following form: $\exists\, x\, y\, z\; P(x,z) \land P(y,z)$, $\mathcal{T} = \{A \sqsubseteq \exists R\}$ and $\mathcal{A} = \{A(a), A(b)\}$.

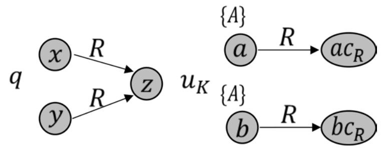


Fig. 8 The graph of query and universal mode

Based on the definition of f_ρ, we have that $(z, x \cdot z, R) \in f_\rho$ and $(z, y \cdot z, R) \in f_\rho$. If $\mathcal{U}_\mathcal{K}$ satisfies q under π and all variables are mapped to anonymous individuals, then $\pi(z) = \pi(x) \cdot c_R = \pi(y) \cdot c_R$. However, $\pi(x) \neq \pi(y)$, this creates a contradiction. Thus, there is no mapping where $\mathcal{U}_\mathcal{K}$ satisfies q under this mapping and all variables are mapped to anonymous individuals.

As shown in the following Fig. 8, the universal model is heterogeneous to the query q. Thus, $\mathcal{U}_\mathcal{K} \nvDash q$.

Theorem 3 *For each consistent DL-Lite$_{horn}^{\mathcal{N}}$ KB $\mathcal{K}$ and each Boolean conjunctive query $q = \exists \mathbf{u}\varphi(\mathbf{u})$ which contains a cyclic structure or fork structure, we have that $\mathcal{U}_\mathcal{K}$ satisfies q if only if $\mathcal{U}_\mathcal{K}^n$ satisfies q with $n = |\mathbf{u}|$.*

Proof (⇒) Based on Lemma 2, if $\mathcal{U}_\mathcal{K}$ satisfies q under π and q contains a cyclic structure or fork structure then there are variables in the query that must be mapped to ABox. Then, we can conclude that $\mathrm{dep}(q, \pi) \leq |\mathbf{u}|$ based on Lemma 1. We label the max certain path of q as $\sigma = t_0 t_1 \cdots t_{m-1} t_m$ and $R(t_{m-1}, t_m) \in q$. Thus, R is m-step generating with m is $\leq |\mathbf{u}|$. By definition of $\mathcal{U}_\mathcal{K}^n$, we have that for every $R \in q$, if $(a, b) \in R^{\mathcal{U}_\mathcal{K}}$ then $a \in \Delta^{\mathcal{U}_\mathcal{K}^n}$ and $b \in \Delta^{\mathcal{U}_\mathcal{K}^n}$. Thus, $(a, b) \in R^{\mathcal{U}_\mathcal{K}^n}$. Since q is connected, for every $A \in q$, suppose $a \in A^{\mathcal{U}_\mathcal{K}}$, then $(a, *) \in R^{\mathcal{U}_\mathcal{K}}$ or $(*, a) \in R^{\mathcal{U}_\mathcal{K}}$. Thus, $a \in \Delta^{\mathcal{U}_\mathcal{K}^n}$ and $a \in A^{\mathcal{U}_\mathcal{K}^n}$. In conclusion, $\mathcal{U}_\mathcal{K}^n \vDash q$.

(⇐) Because $\mathcal{U}_\mathcal{K}^n$ satisfies q and $\mathcal{U}_\mathcal{K}^n$ is a subset of $\mathcal{U}_\mathcal{K}$, so $\mathcal{U}_\mathcal{K}$ satisfies q. □

4 *n*-step Universal Model in OWL 2 DL

OWL 2 DL is based on $\mathcal{SROIQ}$. Because $\mathcal{SROIQ}$ has more complex constructors than *DL-Lite$_{horn}^{\mathcal{N}}$*, more complex information about the described domain can be captured by OWL 2 DL. Particularly, OWL 2 DL has complex restrictions, such as property restrictions and arbitrary cardinality. Besides, UNA is not adopted by OWL 2 DL, that is, different individual names may point to the same individual.

However, with the improvement of ontology language expressiveness, the complexity of reasoning algorithms will also increase. For instance, when dealing with the axioms in the form of UNION, such as $C \sqsubseteq C_1 \sqcup C_2$, it is necessary to try all possibilities and backtrace the tree when

Table 1 OWL 2 DL rewriting rules

No.	TBox axiom	Rewriting TBox axiom
1	$\bigsqcap_{i=1}^{n} C_i \equiv C$	$\bigsqcap_{i=1}^{n} C_i \sqsubseteq C, C \sqsubseteq \bigsqcap_{i=1}^{n} C_i$
2	$C \sqsubseteq \bigsqcap_{i=1}^{n} C_i$	$C \sqsubseteq C_i, 1 \le i \le n$
3	$\exists R.\{a_1, a_2, a_3\} \equiv C$	$\exists R.\{a_1, a_2, a_3\} \sqsubseteq C, C \sqsubseteq \exists R.\{a_1, a_2, a_3\}$
4	$\bigsqcup_{i=1}^{n} C_i \equiv C$	$\bigsqcup_{i=1}^{n} C_i \sqsubseteq C, C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$
5	$\bigsqcup_{i=1}^{n} C_i \sqsubseteq C$	$C_i \sqsubseteq C, 1 \le i \le n$
6	$\mathrm{Dis}(A, B)$	$A \sqsubseteq \neg B, B \sqsubseteq \neg A$
7	$\mathrm{Dis}(r, s)$	$r \sqsubseteq \neg s, s \sqsubseteq \neg r$
8	$r \equiv s$	$r \sqsubseteq s, s \sqsubseteq r$

Table 2 Approximation rules

No.	TBox axiom	Approximation axiom
9	$C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$	$\bigsqcap_{i=1}^{n} \neg C_i \sqsubseteq \neg C$
10	$C \sqsubseteq \exists R.\{a_1, a_2, a_3\}$	$C_1 \equiv \{a_1, a_2, a_3\}, \forall R.\neg C_1 \sqsubseteq \neg C$
11	$C \sqsubseteq \le mR.C_1$	$\ge (m+1)R.C_1 \sqsubseteq \neg C$

there are contradictions in the leaf nodes. This process significantly increases the time and memory consumption of materialization.

To enjoy the high expressiveness of OWL 2 DL and improve materialization efficiency, we implement the support for OWL 2 DL through approximation and rewriting mechanisms. Given an OWL 2 DL ontology, we first attempt to rewrite it as an equivalent $DL\text{-}Lite_{horn}^{\mathcal{N}}$ TBox axiom, if possible. Otherwise, we have three choices, approximate processing or adding additional data structures or other ABox transformation rules. The last two methods can preserve the semantics that approximate processing would lose.

4.1 TBox Transformation

TBox transformation is presented in Tables 1 and 2 (r and s denote role names). Given a $\mathcal{SROIQ}$ TBox axiom, we first rewrite it to an equivalent one according to the rewriting rules in Table 1, also known as normalization in [25]. We add concept and its complement to a complement table (CT), which is designed to record complement semantic.

Because of the normalized TBox axioms $\mathcal{T}$ beyond the expressiveness of $DL\text{-}Lite_{horn}^{\mathcal{N}}$, and the axiom in the form of $C \sqsubseteq \sqcup_{i=1}^{n} C_i$ or $C \sqsubseteq \exists R.\{a_1, a_2, a_3\}$ will lead to non-determinism, we syntactically approximate partial axioms by their complement, as shown in Table 2. Specially, we construct a new concept for nominals at No.10 approximation rule. All TBox transformation rules are sound, as shown in [25].

4.2 ABox Transformation

The semantics of $DL\text{-}Lite_{horn}^{\mathcal{N}}$ do not cover the axiom shown in Table 3. Thus, we design tractable ABox transformation rules, i.e., ABox reasoning rules for them based on the semantics of $\mathcal{SROIQ}$.

An *extended TBox* $\mathcal{T}^*$ is a set of axioms obtained from $\mathcal{T}$ by applying TBox transformation rules and adding ABox transformation rules. Let $\mathcal{K}^* := (\mathcal{T}^*, \mathcal{A})$. Let n be a natural number. The n-step universal model of $(\mathcal{T}^*, \mathcal{A})$ is called an *extended n-step universal model*, denoted by $\mathcal{U}_{\mathcal{K}^*}^{n}$, of $\mathcal{K}$.

By Theorem 2 and the definition of the transformation rules, we can conclude:

Proposition 1 (Approximation) *Let* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *be a consistent KB and* $q = \exists \mathbf{u} \varphi(\mathbf{u}, \mathbf{v})$ *be a rooted CQ with* $|\mathbf{u}| = n$. *For every* $\mathbf{a} \subseteq \mathrm{Ind}(\mathcal{A})$ *with* $|\mathbf{a}| = |\mathbf{v}|$, *if* $\mathbf{a} \in \mathrm{ans}(q, \mathcal{U}_{\mathcal{K}^*}^{n})$ *then* $\mathbf{a} \in \mathrm{cert}(q, \mathcal{K})$.

Example 10 Let $\mathcal{K} = (\{\alpha_1, \alpha_2, \alpha_3\}, \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5\})$ be a KB shown in the following table. GraduateStudent was abbreviated as GS.

Axiom	Expression	Axiom	Expression
α_1	$\mathrm{GS} \equiv \mathrm{Person} \sqcap \ge 3\mathrm{takeCourse}$	β_2	$\mathrm{takeCouse}(b, c_1)$
α_2	$\mathrm{Person} \equiv \mathrm{Woman} \sqcup \mathrm{Man}$	β_3	$\mathrm{takeCouse}(b, c_2)$
α_3	$\mathrm{Dis}(\mathrm{Woman}, \mathrm{Man})$	β_4	$\mathrm{takeCouse}(b, c_3)$
β_1	$\mathrm{GS}(a)$	β_5	$\mathrm{Man}(b)$

First step: we get new TBox axioms:

Table 3 ABox transformation

No.	TBox axiom	ABox reasoning rules
12	$\{C \equiv \{a_1, a_2, \cdots, a_n\}\}$	$\mathcal{A} = \mathcal{A} \cup \{C(a_i)\}, 1 \le i \le n$
13	$\mathrm{Fun}(r)$	$r(a,b) \in \mathcal{A} \wedge r(a,c) \in \mathcal{A} \wedge b \not\doteq c \notin \mathcal{A} \rightarrow \mathcal{A} = \mathcal{A} \cup \{b \doteq c\}$
14	$\mathrm{Trans}(r)$	$r(a,b) \in \mathcal{A} \wedge r(b,c) \in \mathcal{A} \wedge r(a,c) \notin \mathcal{A} \rightarrow \mathcal{A} = \mathcal{A} \cup r(a,c)$
15	$\mathrm{Sym}(r)$	$r(a,b) \in \mathcal{A} \wedge r(b,a) \notin \mathcal{A} \rightarrow \mathcal{A} = \mathcal{A} \cup r(b,a)$

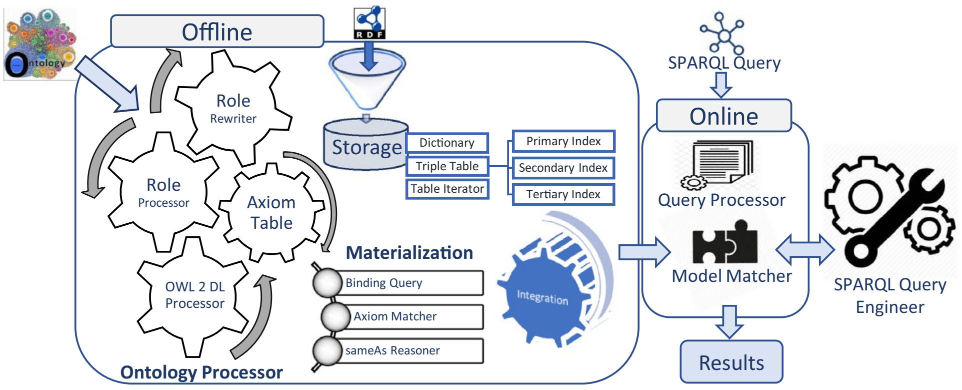


Fig. 9 The architecture of SUMA

α_1^1	GS $\sqsubseteq$ Person	$(\alpha_1, \text{No1}, \text{No2})$
α_1^2	GS $\sqsubseteq \geq$ 3takeCourse.Thing	$(\alpha_1, \text{No1}, \text{No2})$
α_1^3	Person $\sqcap$ $\geq$ 3takeCourse $\sqsubseteq$ GS	$(\alpha_1, \text{No1})$
α_2^1	Woman $\sqsubseteq$ Person	$(\alpha_2, \text{No4}, \text{No5})$
α_2^2	Man $\sqsubseteq$ Person	$(\alpha_2, \text{No4}, \text{No5})$
α_3^1	Woman $\sqsubseteq$ $\neg$Man	$(\alpha_3, \text{No6})$
α_3^2	Man $\sqsubseteq$ $\neg$Woman	$(\alpha_3, \text{No6})$

Second step: from the above axioms, we can get the following new facts:

β_1^1	Person(a)	(β_1, α_1^1)	β_1^2	takeCourse(a, a_1)	(β_1, α_1^2)
β_5^1	Person(b)	(β_5, α_2^2)	β_1^3	takeCourse(a, a_2)	(β_1, α_1^2)
β_6	GS(b)	$(\beta_2 - \beta_4, \beta_5^1, \alpha_1^3)$	β_1^4	takeCourse(a, a_3)	(β_1, α_1^2)
β_5^2	$\neg$Woman(b)	(β_5, α_3^2)	β_8	$(a_1, \dot{\neq}, a_2)$	$(\beta_1^2, \beta_1^3, \alpha_1^2)$
β_7	$(a_1, \dot{\neq}, a_3)$	$(\beta_1^2, \beta_1^4, \alpha_1^2)$	β_9	$(a_2, \dot{\neq}, a_3)$	$(\beta_1^3, \beta_1^4, \alpha_1^2)$

5 The System and Implementation of SUMA

SUMA computes universal model offline and executes queries online. The offline stage consists of three modules: ontology processor, storage, and materialization (Fig. 9).

The ontology processor module has four submodules. OWL 2 DL processor parses the ontology through the OWL API [13], rewrites it to $DL-Lite$ axioms according to rewriting and approximation techniques. It builds the inverse role table and equivalent role table to support role rewriting algorithm.

The role processor implements role scoring algorithm [27]. The role rewriter firstly generates equivalent and inverse role mappings following [27]. For instance, if role d is equivalent to role e and $d.score < e.score$, then d is mapped to e. Formally, $mapping(d) = e, mapping(e) = e$. If $d.score = e.score$ and $d.id > e.id$, then d is mapped to e. Secondly, the role rewriter rewrites axioms and facts.

Example 11 We use the ontologies and facts in Fig. 10 to illustrate the process of role rewriting. The role graph is generated by role score algorithm. Based on the role graph, taughtBy is mapped to teach and like is mapped to love. After choosing the mapping, the role rewriting can be divided into three parts: axiom rewriting, fact rewriting and optimizing rewriting. In fact rewriting phase, taughtBy(Physics, Lisa) and like(Lisa, basketball) are changed to teach(Lisa, Physics) and love(Lisa, basketball), respectively. The axiom range(taughtBy, Teacher) is rewritten at axiom rewriting phase. It was first rewritten to range(teach$^-$, Teacher) and then optimized to domain(taughtBy, Teacher). During the materialization process, taughtBy(Physics, Lisa) and like(Lisa, basketball) are not stored in memory, and Teacher(Lisa) can only be generated once by teach(Lisa, Physics).

The storage module uses the Jena API to load RDF data and generates a dictionary by encoding each RDF resource in integer ID. The dictionary is shared by the storage module and ontology processor. The RDF data are stored as

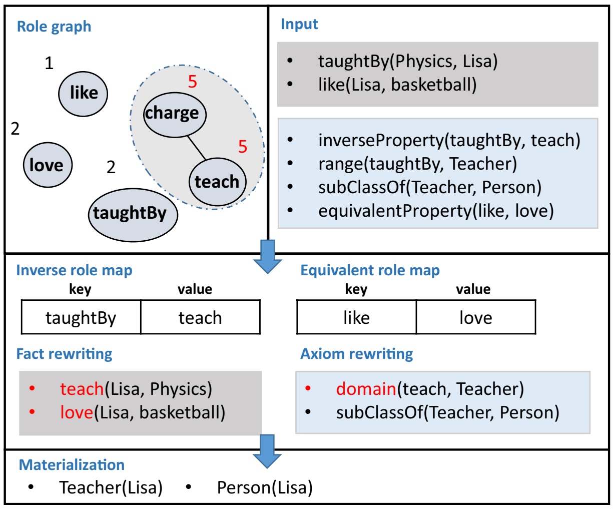


Fig. 10 An example of role rewriting algorithm

a triple table with three types of indexes, e.g., a primary index, a secondary index, and a tertiary index. TableIterator can traverse the triple table efficiently by three types of indexes. And, it maintains an index array that records the n-step model corresponding index ranges in the triple table.

The materialization module has three submodules: binding query, axiom matcher, and sameAs reasoner. The detailed materialization algorithm is shown in Algorithm 1. It iteratively reads a new triple F from the triple table through TableIterator. If F has an equivalent triple G (For instance, $F=(d,R,e)$, $G=(d',R,e')$, $d\doteq d'$, $e\doteq e'$), then the program does not process F to improve reasoning efficiency. The reasoning of F can be divided into three situations.

Algorithm 1 Materialization Algorithm

```
Input: $\mathcal{I}$ : a collection of facts, $\mathcal{T}$: a collection of axioms
Output: $\mathcal{I}$: a collection of expanded facts
1: while $F = \mathcal{I}$.next and $F \neq \varepsilon$ do
2:     $G = \Omega$.getSameAsMapping($F$);
3:     if $G$.equals($F$) then
4:         if $F$ is the form of $(d, \doteq, e)$ then
5:             sameAsReasoning($F$);
6:         else
7:             if $F$ is the form of $(d, P, e) \wedge P$.isFunctionalProperty then
8:                 for $e^* \in \mathcal{I}$.getIndividual($d$, $P$) do
9:                     if $\mathcal{I}$.contains($(e, \dot{\neq}, e^*)$) then
10:                        result in contradiction;
11:                    else $\mathcal{I}$.add($(e, \doteq, e^*)$);
12:                    end if
13:                end for
14:            else
15:                for each $(type, r, F) \in$ matchAxiom($F$, $\mathcal{T}$) do
16:                    for $F' \in \mathcal{I}$.evaluate($type$, $r$, $F$) do
17:                        $\mathcal{I}.add(F')$;
18:                    end for
19:                end for
20:            end if
21:        end if
22:    end if
23: end while
```

Firstly, if F is the form of $(d,\doteq,e)$, it is processed by Algorithm 2. The sameAs reasoning function puts the individual d and e into an equivalent pool and selects the individual with the smallest ID as the identifier. We set the sameAs mapping of d as c if there exists $(d,\doteq,c)$ and c is the smallest ID of the equivalent pool. Secondly, if the role in F is a functional or an inverse functional role, (for instance, $F=(d,P,e)$, and P is a functional role), then, all triples like $(d,P,*)$ are returned by $\mathcal{I}$.getIndividual(d,P). A new fact $(e,\doteq,*)$ is added to $\mathcal{I}$ if $\mathcal{I}$ does not contain a fact $(e,\dot{\neq},*)$. Thirdly, axiom matcher returns all axioms that can match the triple F through matchAxiom$(F,\mathcal{T})$. Binding query function converts these partially matched axioms into partial binding queries. The function $\mathcal{I}$.evaluate executes these queries over $\mathcal{I}$ and returns a new fact.

Algorithm 2 sameAs Reasoning Algorithm

```
Input: $\mathcal{I}$ : $d$, $e$ : individual name, $pool$: a list of equivalent pool
1: idx = mergeEquivalentPool($d$, $e$);
2: $c$ = selectNewIdentifier($pool$[idx]);
3: for $i \in pool$[idx] do
4:     $i$.setSameAsMapping($c$);
5: end for
```

The online part includes a SPARQL processor and a model matcher. The SPARQL processor applies QAA technologies to compute the step size (n) of the model. In essence, it takes *O(1)* time complexity to calculate the number of quantified variables. The model matcher takes n as input and passes the n-step universal model to the SPARQL query engine. The SPARQL query engine executes SPARQL queries and returns query results.

6 Experiments and Evaluations

SUMA delegates SPARQL queries to RDF-3X [24] at this experiment. The experimental environment is a 24-core machine that is equipped with 180GB RAM and Ubuntu 18.04. The experiment is divided into three parts: the evaluation of query answering system on finite model, the evaluation of query answering system on infinite model and role rewriting algorithm evaluation.

Evaluation criteria We test two aspects, (i) the soundness and completeness of the answers, (ii) the scalability of the query answering system. The first aspect is testing the number of queries that the system can answer correctly under the certain answer semantic. The evaluation of the scalability of the query answering system is to test the pre-processing time, consists of data load time and materialization time, and the query processing time on the increasing datasets.

Data load time: This time includes all the data pre-processing steps before materialization, such as constructing a dictionary, generating an index, etc.
Materialization time: The time taken by reasoner to compute consequences.
Query processing time: The time taken by the system to execute a query on the extended data and return the query results.

Comparison system We compare SUMA with the following four systems, all of which use materialization methods.

Pellet is sound and complete in OWL 2 DL. It is adopted as the criterion for soundness and completeness evaluation.

Table 4 The information of datasets

Data	Expressivity	Axioms	Facts	Queries
LUBM(n)	EL++	243	n* 10^5	24
UOBM(n)	$\mathcal{SHION}(\mathcal{D})$	502	2.6n* 10^5	15
DBPedia+	$\mathcal{SHION}(\mathcal{D})$	3000	2.6* 10^7	1024
YAGO	EL++	484,998	1.3* 10^7	260

Table 5 The quality of the answers

Solved	SUMA	Pellet	PAGOdA	gOWL
LUBM(1)	24	24	24	14
LUBM(100)	24	*	24	*
UOBM(1)	15	15	15	2
UOBM(100)	15	*	15	*
DBPedia+	1024	*	1024	*
YAGO	260	*	260	*

PAGOdA employs RDFox for highly scalable reasoning. Therefore, we mainly test the scalability of SUMA with PAGOdA.
gOWL also adopts partial materialization algorithm to solve infinite materialization problem. We evaluate two kinds of partial materialization algorithms from the experimental perspective.
SUMA-N indicates a query answering system that does not use the role rewriting algorithm. SUMA-N is used to evaluate the performance of the role rewriting algorithm.

6.1 Query Answering over Finite Universal Model

Table 4 gives a summary of all datasets and queries used in this experiment. Besides the 14 standard queries of LUBM, we also test ten queries from PAGOdA. The DBPedia [4] axiom is simple. It could be captured by OWL 2 RL [20]. Therefore, we adopt the DBPedia+ axiom and 1024 DBPedia+ queries provided by PAGOdA. The DBPedia+ axiom includes additional tourism ontologies. We generate 260 atomic queries for the YAGO [29] dataset.

6.1.1 The Soundness and Completeness Evaluation

Because Pellet and gOWL cannot give query results on LUBM(100), UOBM(100), DBPedia+ and YAGO in 2 h, we do not display the results of them. As shown in Table 5, SUMA can correctly answer all queries on each test dataset.

6.1.2 The Scalability Test

Since gOWL cannot materialize DBPedia+ and YAGO in less than 2 hours, we compare SUMA and gOWL on small datasets before comparing SUMA and PAGOdA. gOWL takes 1.77 h to materialize LUBM(10), while SUMA takes 1.51 s. The materialization time of gOWL on UOBM(10) is 3.19 h. SUMA only costs 8.34 s materializing UOBM(10). SUMA is more scalable than gOWL.

Next, we test SUMA and PAGOdA on a series of datasets. For LUBM, we use datasets of increasing size with a step of 200. Since UOBM ontology is more complicated than LUBM ontology, we set the UOBM dataset growth step length as 100. For each dataset and ontology, we test the pre-processing time (*pre-time*), data load time, materialization time (*mat-time*), and average query processing time (*avg-time*).

Pre-processing Time Evaluation As shown in Fig. 11, SUMA significantly reduces pre-processing time. Time increases linearly with the size of the dataset. On each test dataset, the pre-processing time of SUMA is faster than PAGOdA.

SUMA only takes 124s to materialize LUBM(1000). The pre-processing time of SUMA on LUBM(1000) is 549s, faster than PAGOdA's 1692s. The time taken by SUMA to materialize UOBM(500) is 411s. The total pre-processing time is 862s. Compared with the 5937s pre-processing time of PAGOdA, SUMA is much faster. SUMA takes 18s and 6s to materialize DBPeida+ and YAGO, respectively. The pre-processing time of SUMA on DBPedia+ and YAGO is 71s and 63s, respectively. PAGOdA costs 309s pre-processing DBPedia+ and 139s pre-processing YAGO. SUMA is more scalable than PAGOdA.

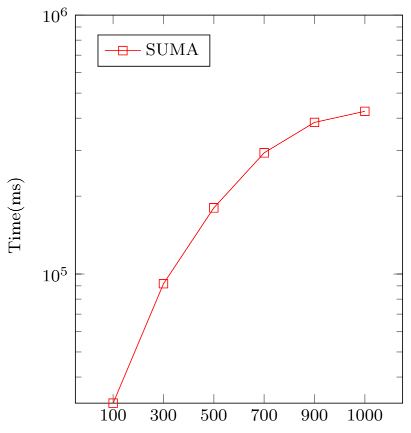
Average Query Processing Time Evaluation

The average query processing time of SUMA on LUBM (1) and UOBM (1) is four and five orders of magnitude faster than Pellet, respectively. Because SUMA and gOWL both rely on existing query engines to perform queries, we only compare SUMA with PAGOdA in the average query processing time evaluation.

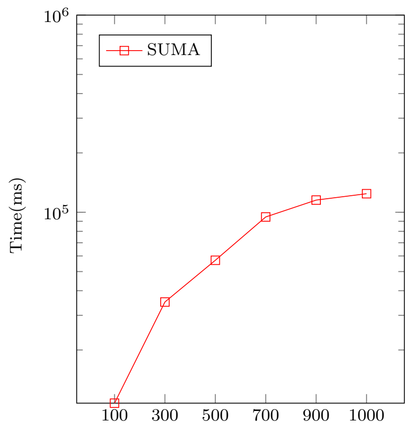
We test the average query processing time of 24 LUBM queries on six LUBM datasets, 15 UOBM queries on five UOBM datasets, 1024 DBPedia+ queries on one DBPedia+ dataset and 260 YAGO queries on the YAGO dataset. As shown in Fig. 12a, SUMA has a faster average query processing time than PAGOdA on all LUBM datasets except LUBM(100). (Time(SUMA, LUBM(100)) = 0.62 s, Time(PAGOdA, LUBM(100)) = 0.57 s). The significant decrease in the query processing time of SUMA on LUBM (500) is related to RDF-3X. RDF-3X can provide shorter query time on larger data by building different efficient indexes.

Figure 12b shows the average query processing time of SUMA is an order of magnitude faster than that of PAGOdA on all UOBM datasets.

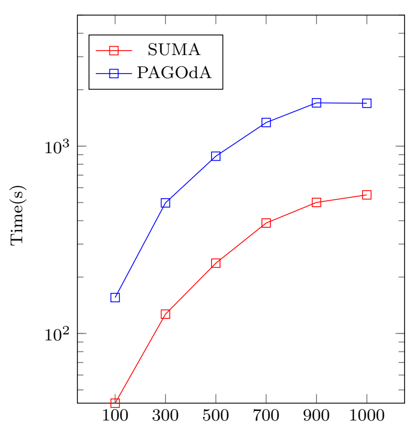
The average query processing time of SUMA on DBPedia+ is 24.337ms faster than PAGOdA's 33.235ms. The



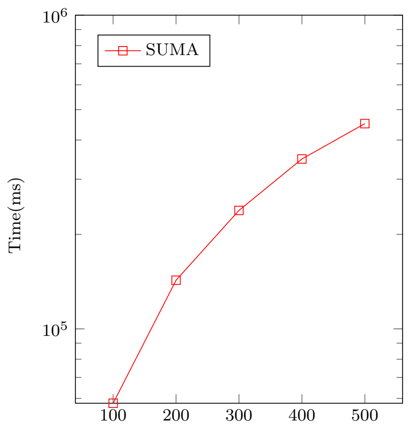
(a) LUBM Load Time



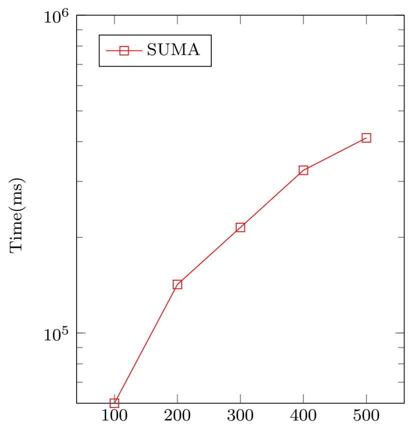
(b) LUBM Mat-time



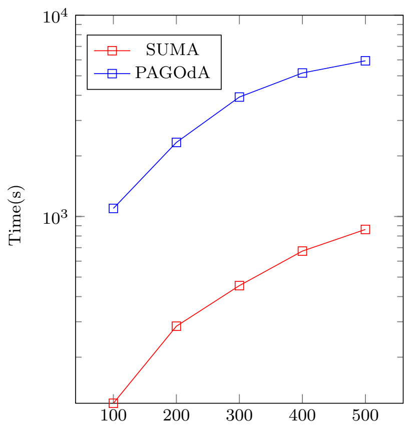
(c) LUBM Pre-time



(d) UOBM Load Time



(e) UOBM Mat-time



(d) UOBM Pre-time

Fig. 11 Pre-processing experimental results

Table 6 The information of datasets

Data	Expressivity	Axioms	Facts	Queries
LUBM+(n)	EL++	245	$n* 10^5$	33
UOBM+(n)	$\mathcal{SHION}(\mathcal{D})$	504	$2.6n* 10^5$	20
LUBM++(n)	EL++	545	$n* 10^5$	*
UOBM++(n)	$\mathcal{SHION}(\mathcal{D})$	805	$2.6n* 10^5$	*

Table 7 The information of queries

Query	♯total	♯cyclic	♯quantified ≥ 3
LUBM+	33	2	9
UOBM+	20	2	3

average query processing time of SUMA on YAGO is 39.166ms faster than PAGOdA's 67.096ms.

6.2 Query Answering over Infinite Universal Model

Since the LUBM, UOBM, DBPedia+ all have a finite universal model, they are not suitable for the second experiment. We add some manual CEQ axioms to the LUBM and UOBM ontologies, respectively. Table 6 shows all the data used at the infinite model evaluation.

We also customize some additional queries to test LUBM+ and UOBM+. Table 7 summarizes our queries. Besides the queries included in the first experiment, we add nine queries for LUBM+ and five customized queries for UOBM+. The third column shows the number of queries that contain a cyclic structure. The number of queries with more than two quantified variables is given in the table's fourth column.

Table 8 The quality of the answers

Solved	SUMA	Pellet	PAGOdA	gOWL
LUBM+(1)	33	33	28	15
UOBM+(1)	20	20	17	3

Table 9 The information of the datasets

Ontology	♯equivalent role	♯inverse role
LUBM	0	4
UOBM	2	8
DBPedia+	0	2
YAGO	0	0

6.2.1 The Soundness and Completeness Evaluation

As shown in Table 8, SUMA and Pellet can calculate all the correct answers for all test queries, whereas PAGOdA is incomplete on five LUBM+ queries (Q2, Q4, Q5, Q6, Q7) and three UOBM+ queries (Q1, Q2, Q3).[1] gOWL is complete only on a few queries.

6.2.2 The Scalability Test

According to statistical analysis of the actual SPARQL queries, more than 96% of the queries include up to 7 triple patterns [10]. Therefore, in most cases, we only need to consider the step of universal model (n) is not greater than 7. Besides, we find that SUMA is also efficient when n is more than 7.

SUMA shows high scalability on LUBM+ and UOBM+. The average query processing time of SUMA on LUBM+(1) and UOBM+(1) is 1.99 ms and 6.48 ms, respectively. It is faster than the PAGOdA's 11.78 ms and 10.40 ms and five orders of magnitude faster than Pellet.

We focus on testing the materialization time of the infinite universal model. To make our test more challenging, we manually add 100 CEQ axioms to LUBM+ and UOBM+ ontologies, named as LUBM++ and UOBM++, as shown in Table 6.

The materialization time of the 15-step universal model of LUBM++(1000) and UOBM++(500) is 276.869 s and 584.600 s, respectively. When $n = 7$, the materialization time of LUBM++(1000) and UOBM++(500) is 172.308 s and 486.504 s, respectively. SUMA is highly scalable on the infinite universal model.

6.3 The Role Rewriting Algorithm Evaluation

As shown in the previous two experiments, SUMA is complete on all test queries, which shows that the role rewriting algorithm does not lose the completeness of materialization. Table 9 shows the number of equivalent and inverse roles in the test dataset. Because the YAGO dataset does not include equivalent roles and inverse roles, it is not used to evaluate role rewriting algorithm.

Materialization Efficiency Evaluation As shown in Fig. 12e, f, on all LUBM and UOBM test data sets, the materialization time of SUMA is faster than that of SUMA-N. SUMA takes 124 s to materialize the LUBM(1000) dataset, while SUMA-N takes 202 s to materialize the LUBM(1000) dataset. SUMA takes 411 s to materialize the UOBM(500) dataset, while SUMA-N takes 515 s to materialize the UOBM(500) dataset.

As shown in Fig. 12g, h, on all LUBM++ and UOBM++ test data sets, the materialization time of SUMA is faster than that of SUMA-N. SUMA takes 276 s to materialize the 15-step LUBM++(1000) model, while SUMA-N takes 351 s to materialize the 15-step LUBM++(1000) data set. SUMA took 584 s to materialize the 15-step UOBM++(500) data set, while SUMA-N took 698 s to materialize the 15-step UOBM++ (500) data set.

On the DBPedia+ dataset, SUMA materialization takes 18 s, while SUMA-N takes 20 s.

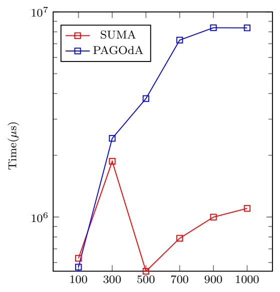
The experiment verifies that the role rewriting algorithm can improve materialization efficiency without reducing the answers' quality.

6.4 Memory Optimization Evaluation

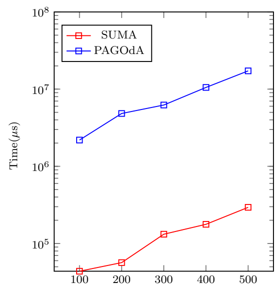
Since SUMA is calculated based on memory, this experiment uses the number of triples in the materialization process to measure the memory consumption of the system. Figure 12i, j shows the number of redundant facts reduced by SUMA. It can be seen from the figure that as the size of the dataset increases, the number of redundant data reduced by the role rewriting algorithm increases linearly. The role rewriting algorithm reduces the memory consumption of LUBM data by 9.48% on average, and reduces the memory consumption of UOBM data by 12.42%. And with the increase in the number of equivalent roles and inverse roles, the effect of memory optimization becomes more obvious.

[1] https://github.com/SUMA-2019/SUMA.

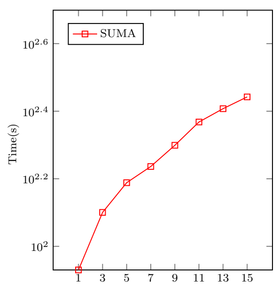
Fig. 12 Experimental results



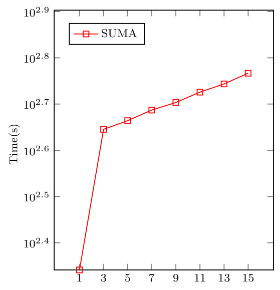
(a) LUBM Avg-time



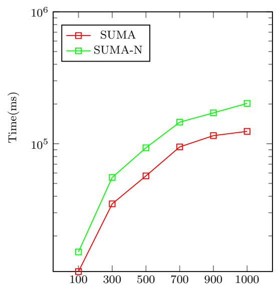
(b) UOBM Avg-time



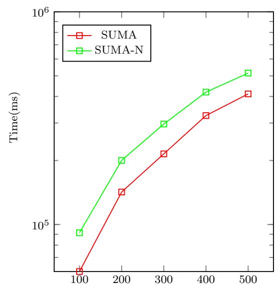
(c) LUBM++ Mat-time



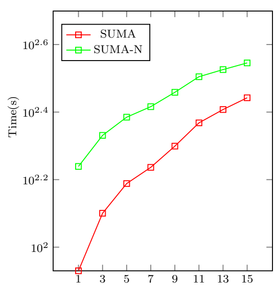
(d) UOBM++ Mat-time



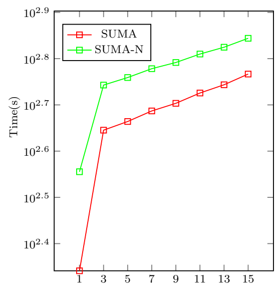
(e) LUBM Mat-time



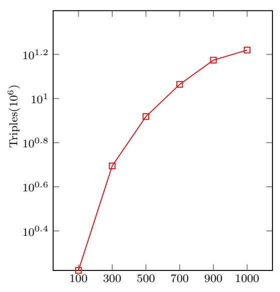
(f) UOBM Mat-time



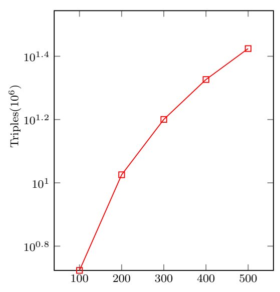
(g) LUBM++(1000) Mat-time



(h) UOBM++(500) Mat-time



(i) LUBM reduced triples



(j) UOBM reduced triples

7 Conclusions

In this paper, we have proposed a partial materialization-based approach for ontology-mediated query answering over OWL 2 DL. Our technique's core idea is that for a rooted conjunctive query or a Boolean conjunctive query with n quantified variables, the answer to the n-step universal model is the same as the answer to the universal model in *DL*. SUMA significantly reduces offline materialization costs by building efficient indexes for facts and rules and integrates role rewriting algorithm. The low complexity materialization algorithm makes SUMA can support efficient reasoning of large-scale datasets. In future works, we are interested in extending this proposal to support distributed reasoning, and extending our approach to support other normalized ontology models.

Acknowledgements We thank Guilin Qi for valuable comments. This work is supported by the National Key Research and Development Program of China (2017YFC0908401), the National Natural Science Foundation of China (61972455, 61672377), and Shenzhen Science and Technology Foundation (JCYJ2017081609 3943197).

References

1. Qin X, Zhang X, Yasin MQ, Wang S, Feng Z, Xiao G (2020) A partial materialization-based approach to scalable query answering in OWL 2 DL. In: International conference on database systems for advanced applications, pp 171–187
2. Artale A, Calvanese D, Kontchakov R, Zakharyaschev M (2009) The DL-Lite family and relations. J Artif Intell Res 36:1–69. https://doi.org/10.1613/jair.2820
3. Bienvenu M (2016) Ontology-mediated query answering: harnessing knowledge to get more from data. In: Proceedings of the twenty-fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, pp. 4058–4061. http://www.ijcai.org/Abstract/16/600
4. Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S (2009) DBpedia—a crystallization point for the web of data. J Web Semant 7(3):154–165. https://doi.org/10.1016/j.websem.2009.07.002
5. Botoeva E, Calvanese D, Santarelli V, Savo DF, Solimando A, Xiao G (2015) Beyond OWL 2 QL in OBDA: rewritings and approximations (extended version). CoRR arxiv:abs/1511.08412
6. Calvanese D, Cogrel B, Komla-Ebri S, Kontchakov R, Lanti D, Rezk M, Rodriguez-Muro M, Xiao G (2017) Ontop: answering SPARQL queries over relational databases. Semant Web 8(3):471–487. https://doi.org/10.3233/SW-160217
7. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Poggi A, Rodriguez-Muro M, Rosati R, Ruzzi M, Savo DF (2011) The MASTRO system for ontology-based data access. Semant Web 2(1):43–53. https://doi.org/10.3233/SW-2011-0029
8. Eiter T, Ortiz M, Simkus M, Tran T, Xiao G (2012) Query rewriting for horn-shiq plus rules. In: Proceedings of the twenty-sixth AAAI Conference on Artificial Intelligence, 22–26 July 2012, Toronto, Ontario, Canada. http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4931
9. Guo Y, Pan Z, Heflin J (2005) LUBM: a benchmark for OWL knowledge base systems. J Web Semant 3(2–3):158–182. https://doi.org/10.1016/j.websem.2005.06.005
10. Han X, Feng Z, Zhang X, Wang X, Rao G, Jiang S (2016) On the statistical analysis of practical SPARQL queries. In: Proceedings of the 19th international workshop on Web and Databases, San Francisco, CA, USA, June 26, 2016, p 2. https://doi.org/10.1145/2932194.2932196
11. Hansen P, Lutz C (2018) Computing fo-rewritings in EL in practice: from atomic to conjunctive queries. CoRR arxiv:abs/1804.06907
12. Harris S, Seaborne A (2013) Sparql 1. 1 query language
13. Horridge M, Bechhofer S (2011) The OWL API: a java API for OWL ontologies. Semant Web 2(1):11–21. https://doi.org/10.3233/SW-2011-0025
14. Horrocks I, Tessaris S (2002) Querying the semantic web: a formal approach. In: The Semantic Web—ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, 9-12 June 2002, Proceedings, pp 177–191. https://doi.org/10.1007/3-540-48005-6_15
15. Kontchakov R, Lutz C, Toman D, Wolter F, Zakharyaschev M (2010) The combined approach to query answering in DL-Lite. In: Principles of knowledge representation and reasoning: proceedings of the twelfth international conference, KR 2010, Toronto, Ontario, Canada, 9-13 May 2010. http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1282
16. Lutz C (2008) The complexity of conjunctive query answering in expressive description logics. In: 4th International Joint Conference Automated Reasoning, IJCAR 2008, Sydney, Australia, 12–15 August 2008, Proceedings, pp 179–193. https://doi.org/10.1007/978-3-540-71070-7_16
17. Lutz C, Seylan I, Toman D, Wolter F (2013) The combined approach to OBDA: taming role hierarchies using filters. In: The Semantic Web—ISWC 2013—12th International Semantic Web Conference, Sydney, NSW, Australia, 21–25 October 2013, Proceedings, Part I, pp 314–330. https://doi.org/10.1007/978-3-642-41335-3_20
18. Ma L, Yang Y, Qiu Z, Xie GT, Pan Y, Liu S (2006) Towards a complete OWL ontology benchmark. In: The semantic web: research and applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, 11–14 June 2006, Proceedings, pp 125–139. https://doi.org/10.1007/11762256_12
19. Meng C, Zhang X, Xiao G, Feng Z, Qi G (2018) gowl: A fast ontology-mediated query answering. In: Proceedings of the ISWC 2018 posters & demonstrations, industry and blue sky ideas tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, 8–12 October 2018. http://ceur-ws.org/Vol-2180/paper-41.pdf
20. Motik B, Grau BC, Horrocks I, Wu Z, Fokoue A, Lutz C (2009) OWL 2 web ontology language profiles. W3C recommendation 27:61
21. Motik B, Nenov Y, Piro R, Horrocks I, Olteanu D (2014) Parallel materialisation of datalog programs in centralised,

main-memory RDF systems. In: Proceedings of the twenty-eighth AAAI Conference on Artificial Intelligence, 27–31 July 2014, Québec City, Québec, Canada, pp 129–137. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8505

22. Motik B, Shearer R, Horrocks I (2009) Hypertableau reasoning for description logics. J Artif Intell Res 36:165–228. https://doi.org/10.1613/jair.2811
23. Nenov Y, Piro R, Motik B, Horrocks I, Wu Z, Banerjee J (2015) Rdfox: a highly-scalable RDF store. In: The Semantic Web—ISWC 2015—14th International Semantic Web Conference, Bethlehem, PA, USA, 11–15 October 2015, Proceedings, Part II, pp 3–20. https://doi.org/10.1007/978-3-319-25010-6_1
24. Neumann T, Weikum G (2008) RDF-3X: a risc-style engine for RDF. Proc VLDB Endow 1(1):647–659. https://doi.org/10.14778/1453856.1453927
25. Pan JZ, Ren Y, Zhao Y (2016) Tractable approximate deduction for OWL. Artif Intell 235:95–155. https://doi.org/10.1016/j.artint.2015.10.004
26. Pérez-Urbina H, Motik B, Horrocks I (2009) A comparison of query rewriting techniques for DL-Lite. In: Proceedings of the 22nd international workshop on Description Logics (DL 2009), Oxford, UK, 27–30 July 2009. http://ceur-ws.org/Vol-477/paper_2.pdf
27. Qin X, Zhang X, Feng Z (2020) Optimizing ontology materialization with equivalent role and inverse role rewriting. In: Companion of the 2020 Web Conference 2020, Taipei, Taiwan, 20–24 April 2020, pp 40–41. https://doi.org/10.1145/3366424.3382687
28. Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y (2007) Pellet: a practical OWL-DL reasoner. J Web Semant 5(2):51–53. https://doi.org/10.1016/j.websem.2007.03.004
29. Tanon TP, Weikum G, Suchanek FM (2020) YAGO 4: a reasonable knowledge base. In: The Semantic Web—17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings, pp 583–596. https://doi.org/10.1007/978-3-030-49461-2_34
30. Zhou Y, Grau BC, Nenov Y, Kaminski M, Horrocks I (2015) Pagoda: pay-as-you-go ontology query answering using a datalog reasoner. J Artif Intell Res 54:309–367. https://doi.org/10.1613/jair.4757