Semantic Web 14 (2023) 997–1050
DOI 10.3233/SW-233382
IOS Press

Conjunctive query answering over unrestricted OWL 2 ontologies

Federico Igne *, Stefano Germano and Ian Horrocks
Department of Computer Science, University of Oxford, Parks Road, Oxford, OX1 3QD, United Kingdom
E-mails: federico.igne@cs.ox.ac.uk, stefano.germano@cs.ox.ac.uk, ian.horrocks@cs.ox.ac.uk

Editor: Guilin Qi, Southeast University, China
Solicited reviews: Xiaowang Zhang, Tianjin University, China; three anonymous reviewers

Abstract. Conjunctive Query (CQ) answering is a primary reasoning task over knowledge bases. However, when considering expressive description logics, query answering can be computationally very expensive; reasoners for CQ answering, although heavily optimized, often sacrifice expressive power of the input ontology or completeness of the computed answers in order to achieve tractability and scalability for the problem. In this work, we present a hybrid query answering architecture that combines various services to provide a CQ answering service for OWL. Specifically, it combines scalable CQ answering services for tractable languages with a CQ answering service for a more expressive language approaching the full OWL 2. If the query can be fully answered by one of the tractable services, then that service is used, to ensure maximum performance. Otherwise, the tractable services are used to compute lower and upper bound approximations. The union of the lower bounds and the intersection of the upper bounds are then compared. If the bounds do not coincide, then the "gap" answers are checked using the "full" service. These techniques led to the development of two new systems: (i) RSAComb, an efficient implementation of a new tractable answering service for RSA (*role safety acyclic*) (ii) ACQuA, a reference implementation of the proposed hybrid architecture combining RSAComb, PAGOdA, and HermiT to provide a CQ answering service for OWL. Our extensive evaluation shows how the additional computational cost introduced by reasoning over a more expressive language like RSA can still provide a significant improvement compared to relying on a fully-fledged reasoner. Additionally, we show how ACQuA can reliably match the performance of PAGOdA, a state-of-the-art CQ answering system that uses a similar approach, and can significantly improve performance when PAGOdA extensively relies on the underlying fully-fledged reasoner.

Keywords: CQ answering, OWL 2, ontology approximation, RSA, combined approach

1. Introduction

Conjunctive Query (CQ) answering over Knowledge Bases (KBs) is a crucial reasoning task for many applications. However, when considering expressive Description Logic (DL) languages, query answering is computationally very expensive, even when considering only complexity w.r.t. the size of the data (*data complexity*) [62]. Fully-fledged reasoners oriented towards CQ answering over unrestricted OWL 2 ontologies exist but, although heavily optimized, they are only effective on small to medium datasets. In order to achieve tractability and scalability for the problem, we need to rely on limiting the expressive power of the input ontology or sacrifice the completeness of the computed answers.

*Corresponding author. E-mail: federico.igne@cs.ox.ac.uk.

Query answering procedures have been developed for several fragments of OWL 2 for which CQ answering is tractable with respect to data complexity [9]. Three such fragments have been standardized as *OWL 2 profiles*, and CQ answering techniques for these fragments have been shown to be highly scalable at the expense of expressive power [10,46,51,68,72,73]. An interesting fragment of OWL 2, tractable for standard reasoning tasks, is RSA, an ontology language that subsumes all the OWL 2 profiles, first presented by Carral et al. [14] and for which a CQ answering algorithm based on the *combined approach* technique [46,47] was proposed by Feier et al. [22].

In order to deal with more expressive ontologies, several techniques have been proposed to compute a *sound subset* of answers to a given CQ. One such technique is to approximate the input ontology to a tractable fragment, so a tractable algorithm can then be used to answer CQs over the approximated ontology. A particularly interesting approach to CQ answering over unrestricted OWL 2 ontologies, using a combination of the aforementioned techniques, is adopted by PAGOdA [85]. Its "pay-as-you-go" approach uses a Datalog reasoner to handle the bulk of the computation, computing lower and upper approximations of the answers to a query, while relying on a fully-fledged OWL 2 reasoner (HermiT [24]) only as necessary to fully answer the query.

While PAGOdA is able to avoid the use of a fully-fledged OWL 2 reasoner in some cases (i.e., when the lower and upper answer approximations coincide), its performance rapidly deteriorates when the input query requires (extensive) use of the underlying OWL 2 reasoner. The computation of lower and upper bounds is achieved by under- and over-approximating the ontology into the $\mathcal{RL}$ profile of OWL 2 so that a tractable reasoner can be used for CQ answering. The tractability of $\mathcal{RL}$ is achieved by avoiding problematic interactions between axioms that can cause an exponential blow-up of the computation (so-called *and-branching*). As it turns out, this elimination of problematic interactions between axioms is rather coarse, and PAGOdA ends up falling back to the underlying OWL 2 reasoner even when it is not really needed.

This work borrows from this "pay-as-you-go" technique and builds upon existing CQ answering techniques over OWL 2 ontologies. We propose a new hybrid query answering architecture that combines black-box[1] services to provide a CQ answering service for OWL. Specifically, it combines scalable CQ answering services for tractable languages with a CQ answering service for a more expressive language approaching the full OWL 2. If the query can be fully answered by one of the tractable services, then that service is used. Otherwise, the tractable services are used to compute lower and upper bound approximations, taking the union of the lower bounds and the intersection of the upper bounds. If the bounds do not coincide, then the "gap" answers are checked using the "full" service. When considering ontology approximations "from below", we introduce a novel algorithm to compute a lower bound to the answers to an input query by means of approximation to RSA. This is done by ensuring that all the constraints for the RSA language are satisfied in the input KB. Similarly, we propose an algorithm to compute an approximation "from above" targetting RSA^{+}, an extension of RSA, for which the combined approach for CQ answering for RSA is still complete. The combined approach for RSA can then be used in both cases to compute the answer bounds. These techniques led to the development of two new system: RSAComb and ACQuA (Answering Conjunctive Queries using Approximation).

RSAComb An efficient implementation [39,41] of the combined approach algorithm for RSA [22], reorganized to fit the new implementation design and the integration of RDFox [54,55,57,60] as a backend Datalog reasoner. We streamlined the execution of the algorithm by factoring out those steps in the combined approach that are *query independent* to make answering multiple queries over the same knowledge base more efficient. In addition, we included an improved version of the filtering step for the combined approach. The system accepts *any* OWL 2 KB and includes a customizable approximation step to languages compatible with the RSA combined approach. The system is further extended with a reference implementation of the novel approximation algorithms for the computation of answer bounds mentioned above.

ACQuA A reference implementation [42] of the hybrid architecture mentioned above, combining RSAComb, PAGOdA [85], and HermiT [24] to provide a CQ answering service for OWL. The resulting system ensures the same "pay-as-you-go" capabilities of the systems it is based on. The system has been designed

[1] By "black-box" we mean that the details of the reasoning process are not relevant, and indeed any reasoner providing comparable reasoning services could be used. However, the semantics are completely transparent, and the results could be interpreted/explained using a wide variety of techniques from the extensive literature on this topic (e.g., [3]). Note that this differs from the definition of the term that is often used, e.g., in the context of Machine Learning (ML), where the semantics of the system is opaque to the user.

to accommodate a high degree of modularity; the services it is built upon can be potentially substituted or augmented with more capable ones to improve the overall performance.

We carried out an extensive evaluation both for RSAComb, as a standalone tool, and for ACQuA, to assess their effectiveness, and compare our results with PAGOdA, aiming, primarily, at improving some shortcomings of the latter tool. Our experimental results show that the new technique yields significant performance improvements in several important application scenarios. Both ACQuA[2] and RSAComb[3] have been released as free and open source software. Source code and documentation are available online.

The present paper includes some previously published work:

- The algorithm for the approximation of an unrestricted OWL 2 ontology to RSA, sound for CQ answering, was presented in [40].
- A full description of RSAComb system was presented in [39].

2. Preliminaries

We assume familiarity with standard concepts of first-order logic (FOL) such as term, variable, constant, predicate, atom, literal; refer to [1,5] for a formal introduction to these concepts.

We define a *rule* as an expression of the form $\varphi(\vec{x}, \vec{y}) \rightarrow \psi(\vec{x})$, with $\varphi(\vec{x}, \vec{y})$ a conjunction of literals over variables $\vec{x} \cup \vec{y}$ and $\psi(\vec{x})$ a non-empty conjunction of atoms over $\vec{x}$. Given a role r, we denote $\texttt{head}(r)$ the set of atoms in $\psi(\vec{x})$, and $\texttt{body}^+(r)$ (resp. $\texttt{body}^-(r)$) the set of *positive* (resp. *negative*) literals in $\varphi(\vec{x}, \vec{y})$.

We will call a rule *definite* without negation in its body, and *Datalog* a function-free definite rule. A Datalog rule is *disjunctive* if it admits disjunction in the head. A *fact* is a Datalog rule with an empty body. The definition can be trivially extended to sets of rules.

A *program* Π is a set of rules. Let $pred(X)$ be the set of predicates in X (with X being either a set of atoms, a rule, or a program). A *stratification* of a program Π is a function $\delta : pred(\Pi) \rightarrow \{1, \ldots, k\}$ with $k \leqslant |pred(\Pi)|$, s.t. for every rule $r \in \Pi$ and $p \in pred(\texttt{head}(r))$ it holds:

- for every $q \in pred(\texttt{body}^+(r))$, $\delta(q) \leqslant \delta(p)$;
- for every $q \in pred(\texttt{body}^-(r))$, $\delta(q) < \delta(p)$;

The *stratification partition* of Π induced by δ is the sequence $(\Pi_1, \ldots, \Pi_k)$ with each Π_i be the set of rules $r \in \Pi$ s.t. $\texttt{max}_{a \in \texttt{head}(r)}(\delta(pred(a))) = i$. Programs Π_i are called *strata* of Π. A program is *stratified* if it admits a stratification. All definite programs are stratified.

A stratified program Π has a *least* Herbrand model (LHM), which is constructed using the immediate consequence operator T_Π. Let $\mathcal{H}_U$ and $\mathcal{H}_B$ be the *Herbrand universe* and the *Herbrand base* of Π. Let $S \in \mathcal{H}_B$, then, $T_\Pi(S)$ consists of all facts in $\texttt{head}(r)\sigma$ with $r \in \Pi$ and σ a substitution for the variables in r to $\mathcal{H}_U$ satisfying $\texttt{body}^+(r)\sigma \subseteq S$ and $\texttt{body}^-(r)\sigma \cap S = \emptyset$. The *powers* of T_Π are defined as follows: (i) $T^0_\Pi(S) = S$, (ii) $T^{n+i}_\Pi(S) = T_\Pi(T^n_\Pi(S))$, (iii) $T^\omega_\Pi(S) = \bigcup_{i=0}^{\infty} T^i_\Pi(S)$. Let $(\Pi_1, \ldots, \Pi_k)$ be a stratification partition for Π. We define $U_1 = T^\omega_{\Pi_1}(\emptyset)$ and for each $1 \leqslant i < k$, $U_{i+1} = T^\omega_{\Pi_{i+1}}(U_i)$ Then, the LHM of Π is U_k and is denoted as $M[\Pi]$.

Given a stratified program Π, we define $\Pi^{\approx,\top}$ the program extended with standard axiomatization rules for equality ($\approx$) and truth value ($\top$) [22].

2.1. Ontologies and conjunctive query answering

Next we give a brief overview of the description logic languages used in the paper. We will define them as restrictions of $\mathcal{SROIQ}$ [36], the description logic underpinning the OWL 2 ontology language [58], standardized by W3C. An ontology *signature* is a triple $\langle N_C, N_R, N_I \rangle$ of computable disjoint sets of *concept names*, *role names*

[2] https://github.com/KRR-Oxford/ACQuA
[3] https://github.com/KRR-Oxford/RSAComb

Table 1
Normalized $\mathcal{SROIQ}$ axioms and their translation into logic rules

Axiom / Role / Assertion α		Definite rules $\pi(\alpha)$
(R1)	R^-	$R(x,y) \rightarrow R^-(y,x) \quad R^-(y,x) \rightarrow R(x,y)$
(R2)	$R \sqsubseteq S$	$R(x,y) \rightarrow S(x,y)$
(R3)	$R \sqcap S \sqsubseteq \bot$	$R(x,y) \wedge S(x,y) \rightarrow \bot$
(R4)	$R \circ S \sqsubseteq T$	$R(x,y) \wedge S(y,z) \rightarrow T(x,z)$
(T1)	$\bigsqcap_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{i=1}^{m} B_i$	$\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{i=1}^{m} B_i(x)$
(T2)	$A \sqsubseteq \{a\}$	$A(x) \rightarrow x \approx a$
(T3)	$\exists R.A \sqsubseteq B$	$R(x,y) \wedge A(y) \rightarrow B(x)$
(T4)	$A \sqsubseteq \leqslant m R.B$	$A(x) \wedge \bigwedge_{i=1}^{m+1}[R(x,y_i) \wedge B(y_i)] \rightarrow \bigvee_{1 \leqslant i < j \leqslant m+1} y_i \approx y_j$
(T5)	$A \sqsubseteq \exists R.B$	$A(x) \rightarrow R(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$
(T6)	$A \sqsubseteq \text{Self}(R)$	$A(x) \rightarrow R(x,x)$
(T7)	$\text{Self}(R) \sqsubseteq A$	$R(x,x) \rightarrow A(x)$
(A1)	$A(a)$	$\rightarrow A(a)$
(A2)	$R(a,b)$	$\rightarrow R(a,b)$

and *individuals* respectively. Two special concepts are provided: $\bot$ (bottom concept) and $\top$ (top concept). We define a *role* as an element of $N_R \cup \{R^- \mid R \in N_R\}$, where R^- is called *inverse role*. We also introduce a function $Inv(\cdot)$ closed for roles s.t. $\forall R \in N_R : Inv(R) = R^-, Inv(R^-) = R$. An *RBox* $\mathcal{R}$ is a finite set of axioms of type (R2)–(R4) in Table 1 with R, S, T roles. We denote $\sqsubseteq^*_{\mathcal{R}}$ as a minimal relation over roles closed by reflexivity and transitivity s.t. $R \sqsubseteq^*_{\mathcal{R}} S, Inv(R) \sqsubseteq^*_{\mathcal{R}} Inv(S)$ hold if $R \sqsubseteq S \in \mathcal{R}$. A role R is *transitive* if a role T exists such that $T \sqsubseteq^*_{\mathcal{R}} R$, $R \sqsubseteq^*_{\mathcal{R}} T$ and either $T \circ T \sqsubseteq T \in \mathcal{R}$ or $T^- \circ T^- \sqsubseteq T^- \in \mathcal{R}$. A *TBox* $\mathcal{T}$ is a set of axioms of type (T1)–(T7) where $A, B \in N_C$, $a \in N_I$, R is a role and $Self(\cdot)$ denotes the *local reflexivity* of a role. An *ABox* $\mathcal{A}$ is a finite set of *assertions* of type (A1)–(A2) with $A \in N_C$, $a, b \in N_I$ and $R \in N_R$. A $\mathcal{SROIQ}$ ontology is a set of axioms $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$.[4] An ontology is $\mathcal{SHOIQ}^+$ if we restrict axioms (R4) to role transitivity (i.e., $R = S = T$). An ontology is $\mathcal{SHOIQ}$ if we further exclude axioms of type (T6), (T7) and (R3). An $\mathcal{ALCHOIQ}^+$ ontology (resp. $\mathcal{ALCHOIQ}$) is obtained from $\mathcal{SHOIQ}^+$ (resp. $\mathcal{SHOIQ}$) by disallowing (R4) axioms altogether. A Horn-$\mathcal{ALCHOIQ}^+$ ontology (resp. Horn-$\mathcal{ALCHOIQ}$) is obtained from $\mathcal{ALCHOIQ}^+$ (resp. $\mathcal{ALCHOIQ}$) by restricting $m = 1$ in axioms (T1) and (T4). Finally, given an ontology language $\mathcal{L}$, we define an $\mathcal{L}$ *knowledge base* as a couple $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ comprising an $\mathcal{L}$ ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$ and an ABox $\mathcal{A}$.

Table 1 also introduces a normal form for each of these description logic languages, and w.l.o.g. we assume that any ontology introduced is restricted to these axioms. Each axiom in Table 1 corresponds to a single logic rule, provided on the right. We define $\pi(\cdot)$ as the translation function from axioms to logic rules; the function can be trivially extended to sets of axioms by mapping π over the set and to knowledge bases (i.e., $\pi(\mathcal{K}) = \{\pi(\alpha) \mid \alpha \in \mathcal{A} \cup \mathcal{T} \cup \mathcal{R}\}$). Furthermore, we define $\Pi_{\mathcal{K}} = \pi(\mathcal{K})^{\bot,\approx}$ as the logic program derived from $\mathcal{K}$ extended with *bottom* and *equality* axiomatization rules (as defined in [14]).

OWL 2 *profiles* [53] have been defined as fragments of OWL 2, designed to provide a desirable balance between computational complexity of standard reasoning tasks and expressiveness of the ontology language. We will define these standard profiles as fragments of Horn-$\mathcal{ALCHOIQ}$.[5]

1. OWL 2 EL is based on the $\mathcal{EL}^{++}$ DL language [4,6]; it does not contain *inverse roles* (R1) and axioms of type (T4);
2. OWL 2 RL is inspired by Description Logic Programs [29] and corresponds to a subset of Datalog; it does not contain axioms of type (T5);
3. OWL 2 QL is based on the DL-Lite$_{\mathcal{R}}$ DL language [10]; it does not contain axioms (T2), (T4), axioms (T1) satisfy $n = 1$ and axioms (T3) satisfy $A = \top$.

[4] In order to achieve decidability of reasoning, $\mathcal{SROIQ}$ ontologies must satisfy certain additional requirements. These, however, do not affect the technical results reported in this paper.

[5] Property chain and transitivity axioms are not taken into consideration here to keep definitions compatible with [22].

Note that, when not considering *transitive roles*, the definition of $\mathcal{EL}^{++}$, the logic underpinning OWL 2 EL, matches $\mathcal{ELHO}^{r}_{\bot}$ [72,85].

A *conjunctive query (CQ)* q is a formula $\exists\vec{y}.\psi(\vec{x},\vec{y})$ with $\psi(\vec{x},\vec{y})$ a *conjunction* of function–free atoms over $\vec{x}\cup\vec{y}$, and $\vec{x}$, $\vec{y}$ are called *answer variables* and *bounded variables*, respectively. We call *boolean conjunctive query (BCQ)* a query where $|\vec{x}|=0$ (i.e., the set of answer variable is empty). A query is *atomic* if $\psi(\vec{x},\vec{y})$ consists of a single atom and $|\vec{y}|=0$.

A knowledge base $\mathcal{K}=\langle\mathcal{T}\cup\mathcal{R},\mathcal{A}\rangle$ is satisfiable if $\Pi_{\mathcal{K}}\not\models\exists y.\bot(y)$. A tuple of constants $\vec{a}$ is a *possible answer* to q w.r.t. $\mathcal{K}$ if $|\vec{a}|=|\vec{x}|$ and each constant in $\vec{a}$ occurs in $\mathcal{K}$. $\vec{a}$ is a *certain answer* to q w.r.t. $\mathcal{K}$ if $\mathcal{K}$ is *unsatisfiable* or $\vec{a}$ is a possible answer and $\Pi_{\mathcal{K}}\models\exists\vec{y}.\psi(\vec{a},\vec{y})$. The set of certain answers to a query q is denoted by $cert(q,\mathcal{K})$. We say that a possible answer $\vec{a}$ is a *ground answer* to q w.r.t. a satisfiable knowledge base $\mathcal{K}$ if a tuple of constants $\vec{e}$ in $\mathcal{K}$ exists such that $|\vec{y}|=|\vec{e}|$ and $\Pi_{\mathcal{K}}\models\psi(\vec{a},\vec{e})$. We denote the set of ground answers with $ground(q,\mathcal{K})$. It is straightforward to see that $ground(q,\mathcal{K})\subseteq cert(q,\mathcal{K})$.

CQs can be alternatively represented as Datalog rules. Let P_q a fresh predicate of arity $|\vec{x}|$ uniquely associated with q. Then let $q_r = P_q(\vec{x}) \leftarrow \psi(\vec{x},\vec{y})$ be the Datalog rule representing q. This allows to characterize certain answers by means of entailment of a single fact, i.e., $\vec{a}\in cert(q,\mathcal{K})$ iff $\Pi_{\mathcal{K}}\cup\{q_r\}\models P_q(\vec{a})$.

We say that q is *internalizable* if it can be turned into an ontology axiom. This process is known as *rolling-up* [36] and is implemented in some solvers (e.g., Pellet [71]) to provide sound and complete CQ answering over OWL 2 DL over *certain answer semantics* when considering internalizable queries.

Given a knowledge base $\mathcal{K}$ and a CQ q, we call *conjunctive query answering* the reasoning problem of computing all certain answers of q w.r.t. $\mathcal{K}$. The decision problem associated with CQ answering is called *conjunctive query entailment* (CQE). Given a knowledge base $\mathcal{K}$, a CQ q and a possible answer $\vec{a}$, CQE is the problem of deciding whether $\Pi_{\mathcal{K}}\models\exists\vec{y}\psi(\vec{a},\vec{y})$.

2.2. PAGOdA

PAGOdA is a hybrid reasoner for sound and complete CQ answering over OWL 2 KBs, adopting a "pay-as-you-go" technique to compute the certain answers to a given query. The idea is to compute lower/upper bound approximations to the answers to a query by approximating the input ontology into a less expressive language and possibly provide a fallback (more expensive) algorithm to process the answers in the gap between the bounds; to achieve this, it uses a combination of a *Datalog reasoner* and a *fully-fledged OWL 2 reasoner*. PAGOdA treats the two systems as black boxes and tries to offload the bulk of the computation to the former and relies on the latter only when necessary.

The capabilities, performance, and scalability of PAGOdA inherently depend on the ability of the fully-fledged OWL 2 reasoner in use, and the ability to delegate the workload to a given Datalog reasoner. In the best scenario, with an OWL 2 reasoner, PAGOdA is able to answer internalisable queries [36] under certain answer semantics [85] over OWL 2 DL.

In the following is a high level description of the procedure adopted by PAGOdA to compute the answers to a query. This will prove useful to understand how this approach will be later integrated in our system, ACQuA. For a more in-depth description of the algorithm and heuristics in use, we refer the reader to [84].

Given a KB $\mathcal{K}=\langle\mathcal{T}\cup\mathcal{R},\mathcal{A}\rangle$[6] and a query q, PAGOdA executes the following steps in order to compute the answers to q w.r.t. $\mathcal{K}$:

1. the Datalog reasoner is exploited to compute a *lower bound* L^q and an *upper bound* U^q to the answers to the query q. This is achieved by approximating the input KB $\mathcal{K}$ into a tractable language to be handled by the Datalog reasoner. Depending on the approximation procedure, running the query over the approximated ontology will result in either a lower or an upper bound of the certain answers to the query. The lower bound L^q is obtained as follows:
   (a) the *disjunctive Datalog* subset of the input ontology, denoted with $\mathcal{K}^{DD}$, is computed by dropping any axiom that does not correspond to a disjunctive Datalog rule;

[6] In the following we consider the input KB to be *consistent* and *normalized*. This is ensured by PAGOdA's preprocessing step.

(b) using a variant of *shifting* [18], $\mathcal{K}^{DD}$ is polynomially transformed in order to eliminate disjunction in the head. The resulting ontology $\texttt{shift}(\mathcal{K}^{DD})$ is sound but not necessarily complete for CQ answering;
(c) a first materialization is performed, i.e., $M_1 = M[\texttt{shift}(\mathcal{K}^{DD})]$, and the resulting facts are added back to the input knowledge base to obtain $\mathcal{K}' = \langle \mathcal{T} \cup \mathcal{R}, \mathcal{A} \cup M_1 \rangle$;
(d) the $\mathcal{ELHO}^r_\bot$ [72] subset of $\mathcal{K}'$ is computed, denoted $\mathcal{K}'_{\mathcal{EL}}$, dropping any axiom that is not in $\mathcal{ELHO}^r_\bot$;
(e) the *combined approach* for $\mathcal{ELHO}^r_\bot$ [51,73] is used to compute the answers to the query q over $\mathcal{K}'_{\mathcal{EL}}$.

The upper bound U^q is computed by executing the query over the ontology, modified as follows:

(a) the $\bot$ concept is substituted with a fresh concept name $\bot_s$ to avoid directly deriving *falsehood*;
(b) *disjuncts* in the head of an axiom are reduced to a single disjunct. The "most favourable" disjunct is chosen according to a polynomial *choice function* that reasons over the dependency graph of the input ontology;
(c) *existential* axioms of type (T5) are *constant Skolemized*.

2. If lower and upper bound coincide (i.e., $L^q = U^q$) then the Datalog reasoner was able to provide a sound and complete set of answers to the input query. The computation terminates;
3. otherwise, the "gap" between the upper and lower bound (i.e., $G^q = U^q \setminus L^q$) is a set of answers that need to be verified against the KB using a fully-fledged OWL 2 reasoner. The Datalog reasoner is again exploited for this step to compute a *subset* $\mathcal{K}^q$ of the KB $\mathcal{K}$ that is enough to check whether the answers in G^q are certain or spurious;
4. for each $\vec{a} \in G^q$, the fully-fledged reasoner is used to check whether $\mathcal{K}^q \models q(\vec{a})$. This process is further optimized by reducing the number of answers in G^q that need to be checked by means of *summarization* [16];
5. once all spurious answers have been removed from G^q, $L^q \cup G^q$ is returned.

Let's take the lower bound computation as an example: the two performed approximations (i.e., to *disjunctive Datalog* and to $\mathcal{ELHO}^r_\bot$) are handled independently, by means of materialization in the first case, and the combined approach in the second; this allows PAGOdA to avoid having to deal with *and-branching* and the resulting intractability of most reasoning problems (see Definition 2.1). In fact, OWL 2 RL (Datalog) and $\mathcal{ELHO}^r_\bot$ are used by PAGOdA to eliminate *all* interactions between axioms (T5) and either axioms (T4) or axioms (T3) and (R1).[7] However, not all such interactions cause an exponential jump in complexity, and PAGOdA's filtering of such cases is unnecessarily coarse. We will see in the next sections, how this procedure can be improved by introducing an alternative approximation algorithm.

PAGOdA's reference implementation[8] uses RDFox as a Datalog reasoner and HermiT as the underlying fully-fledged reasoner. It accepts any OWL 2 DL ontology as input, alongside a dataset in *Turtle* format and CQs in SPARQL [33].

PAGOdA ensures that the returned answers are always *complete* under ground semantics, while being ultimately limited by the capabilities of HermiT when considering the returned answers under certain answer semantics. HermiT does not natively support CQ answering and the process needs to be reduced to fact entailment first. This is possible when the input query is *internalisable*, i.e., the query can be *rolled-up* into a set of DL concept assertions. In this scenario PAGOdA returns a set of answers that is sound and complete under certain answers semantics if the bounds match or the query can be *internalized* into a DL concept. Otherwise, PAGOdA will return a sound set of answers (complete under ground semantics) and a bound on the incompleteness of the computed answers (under certain answers semantics).

2.3. The RSA ontology language

As we mentioned above, ACQuA combines multiple ontology approximation algorithms to compute the answers to an input query. As part of this work, we propose novel approximation algorithms that target RSA (and its extension RSA$^+$). In this section, we provide a brief introduction to the RSA ontology language [14], along with a description of a combined approach for CQ answering [22].

[7] OWL 2 RL does not allow axioms (T5) and $\mathcal{EL}$ (which contains $\mathcal{ELHO}^r_\bot$) does not allow axioms (T4) or inverse roles (R1).
[8] https://github.com/KRR-Oxford/PAGOdA

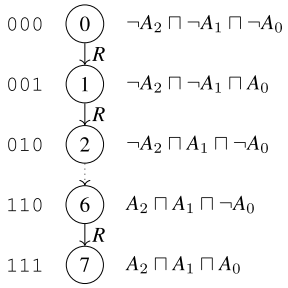


Fig. 1. Example of exponential model enumerating numbers from 0 to $2^n - 1$ for $n = 3$. The KB is polynomial in n.

RSA (*role safety acyclic*) is a class of ontologies designed to subsume all OWL 2 profiles, while maintaining tractability of standard reasoning tasks. The RSA ontology language is designed to avoid interactions between axioms that can result in the ontology being satisfied only by exponentially large (and potentially infinite) models. This problem is often called *and-branching* and can be caused by interactions between axioms of type (T5) with either axioms (T3) and (R1), or axioms (T4), in Table 1.

Example 2.1. Interaction between existential quantifiers (axioms of type (T5)) and universal quantifiers (encoded by axioms of type (T3) and (R1)) can lead to an ontology that may only be satisfied by models of exponential size.

Consider the following knowledge base with ABox $\mathcal{A} = \{(\neg A_0 \sqcap \cdots \sqcap \neg A_{n-1})(a)\}$ for some n, and a TBox containing the following axioms, for $0 \leqslant i < n$:

$$\neg A_i \sqcap \prod_{j<i} A_j \sqsubseteq B_i \sqcap \exists R.A_i \sqcap \forall R.\left(\prod_{j<i} \neg A_j\right) \tag{1}$$

$$\forall_{j>i}(B_i \sqcap A_j \sqsubseteq \forall R.A_j) \tag{2}$$

$$\forall_{j>i}(B_i \sqcap \neg A_j \sqsubseteq \forall R.\neg A_j) \tag{3}$$

The size of the knowledge base is polynomial w.r.t. n. It can be shown that this knowledge base enforces a chain of individual of length 2^n where each individual represents a number from 0 to $2^n - 1$ encoded in binary (i.e., each A_i represents a bit in position i, where an A_i encodes a 1 and a $\neg A_i$ encodes a 0). Figure 1 shows an example of the exponentially large model for $n = 3$.

RSA is based on the Horn-$\mathcal{ALCHOIQ}$ ontology language, restricting the interaction between axioms to ensure a polynomial bound on model size [14]. For the following section we will consider a generic Horn-$\mathcal{ALCHOIQ}$ knowledge base $\mathcal{K} = \langle \mathcal{T} \cup \mathcal{R}, \mathcal{A} \rangle$ over the signature $\Sigma_{\mathcal{K}} = \langle N_C, N_R, N_I \rangle$.

Definition 2.1 ([22], Definition 1)**.** A role R in $\mathcal{K}$ is *unsafe* if it occurs in axioms (T5), and there is a role S s.t. either of the following holds:

1. $R \sqsubseteq^*_{\mathcal{R}} \mathit{Inv}(S)$ and S occurs in an axiom (T3) with left-hand side concept $\exists S.A$ where $A \neq \top$;
2. S is in an axiom (T4) and $R \sqsubseteq^*_{\mathcal{R}} S$ or $R \sqsubseteq^*_{\mathcal{R}} \mathit{Inv}(S)$.

A role R in $\mathcal{K}$ is *safe* if it is not unsafe.

Note that, all OWL 2 profiles ($\mathcal{RL}$, $\mathcal{EL}$ and $\mathcal{QL}$) as defined in Section 2.1, contain only *safe* roles.

Example 2.2. In Example 2.1, R is unsafe. In fact, even for $n = 1$, we have that

$$\neg A_0 \sqsubseteq B_0 \sqcap \exists R.A_0 \qquad B_0 \sqcap A_1 \sqsubseteq \forall R.A_1 \tag{4}$$

are part of the program and can be rewritten as

$$\neg A_0 \sqsubseteq B_0 \qquad \neg A_0 \sqsubseteq \exists R.A_0 \qquad B_0 \sqcap A_1 \sqsubseteq X \qquad \exists R^-.X \sqsubseteq A_1 \tag{5}$$

using some standard normalization and the fact that $A \sqsubseteq \forall R.B \equiv \exists R^- A \sqsubseteq B$.

Then, R appears in an axiom of type (T5), $S \equiv R^-$, $R \sqsubseteq^*_{\mathcal{R}} Inv(S)$ and S occurs in an axiom (T3) with left-hand side concept $\exists S.A$ where $A \neq \top$;

The distinction between safe and unsafe roles makes it possible to strengthen the translation π from Table 1 while preserving satisfiability and entailment of unary facts.

Definition 2.2 ([22], Definition 2)**.** Let $v^A_{R,B}$ be a fresh constant for each pair of concepts A, B and each safe role R in $\mathcal{K}$. The function π_{safe} is defined for each axiom α in $\mathcal{K}$:

$$\pi_{\text{safe}}(\alpha) = \begin{cases} A(x) \rightarrow R(x, v^A_{R,B}) \wedge B(v^A_{R,B}) & \text{if } \alpha \text{ is of type (T5) and } R \text{ safe} \\ \pi(\alpha) & \text{otherwise.} \end{cases} \tag{6}$$

Let $\mathcal{P} = \{\pi_{\text{safe}}(\alpha) \mid \alpha \in \mathcal{K}\}$ and $\mathcal{P}_{\mathcal{K}} = \mathcal{P}^{\approx,\top}$.

Theorem 2.1 ([14], Theorem 2)**.** *A Horn-$\mathcal{ALCHOIQ}$ knowledge base $\mathcal{K}$ is satisfiable iff $\mathcal{P}_{\mathcal{K}} \not\models \bot$. If $\mathcal{K}$ is satisfiable, then, $\mathcal{K} \models A(c)$ iff $A(c) \in M[P_{\mathcal{K}}]$ for each unary predicate A and constant c in $\mathcal{K}$.*

Note that if $\mathcal{K}$ contains unsafe roles, the model $M[\mathcal{P}_{\mathcal{K}}]$ might be exponentially large or infinite.

Potential bad interactions between unsafe roles can be avoided by detecting any cyclic or diamond-shape materialization involving unsafe roles. This check is performed by constant Skolemizing all existential axioms with an unsafe role, and building a graph representing the materialization process, projected on unsafe interaction. Checking that the graph is an oriented forest, along with some additional conditions which preclude harmful interactions between equality-generating axioms and inverse roles, leads to the definition of RSA.

Definition 2.3 ([22], Definition 3)**.** Let $\texttt{PE}$ and $\texttt{E}$ be fresh binary predicates, let $\texttt{U}$ be a fresh unary predicate, and let $u^A_{R,B}$ be a fresh constant for each concept A, $B \in N_C$ and each role $R \in N_R$. Then, for each axiom α in $\mathcal{K}$

$$\pi_{\text{RSA}}(\alpha) = \begin{cases} A(x) \rightarrow R(x, u^A_{R,B}) \wedge B(u^A_{R,B}) \wedge \texttt{PE}(x, u^A_{R,B}) & \text{if } \alpha \text{ is of type (T5)} \\ \pi(\alpha) & \text{otherwise.} \end{cases} \tag{7}$$

The program $\mathcal{P}_{\text{RSA}}$ consists of $\pi_{\text{RSA}}(\alpha)$ for each $\alpha \in \mathcal{K}$, rule $\texttt{U}(x) \wedge \texttt{PE}(x, y) \wedge \texttt{U}(y) \rightarrow \texttt{E}(x, y)$ and facts $\texttt{U}(u^A_{R,B})$ for each $u^A_{R,B}$, with R unsafe.

Let M_{RSA} be the LHM of $\mathcal{P}^{\approx,\top}_{\text{RSA}}$. Then, $G_{\mathcal{K}}$ is the digraph with an edge (c, d) for each $\texttt{E}(c, d)$ in M_{RSA}. Knowledge base $\mathcal{K}$ is *equality-safe* if:

(i) for each pair of atoms $w \approx t$ (with w and y distinct) and $R(t, u^A_{R,B})$ in M_{RSA} and each role S s.t. $R \sqsubseteq Inv(S)$, it holds that S does not occur in an axiom (T4), and

(ii) for each pair of atoms $R(a, u^A_{R,B})$, $S(u^A_{R,B}, a)$ in M_{RSA} with $a \in N_I$, there is no role T such that both $R \sqsubseteq^*_{\mathcal{R}} T$ and $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$ hold.

We say that $\mathcal{K}$ is RSA if it is *equality-safe* and $G_{\mathcal{K}}$ is an oriented forest.[9]

Definition 2.4. With reference to Definition 2.3, let $\mathcal{K}$ be a Horn-$\mathcal{ALCHOIQ}^+$ knowledge base. Then, $\mathcal{K}$ is RSA$^+$ if it is *equality-safe* and $G_{\mathcal{K}}$ is an oriented forest.

The fact that $G_{\mathcal{K}}$ is a DAG ensures that the LHM $M[\mathcal{P}_{\mathcal{K}}]$ is finite, whereas the lack of "diamond-shaped" subgraphs in $G_{\mathcal{K}}$ guarantees polynomiality of $M[\mathcal{P}_{\mathcal{K}}]$. The definition gives us a programmatic procedure to determine whether a Horn-$\mathcal{ALCHOIQ}$ (resp. Horn-$\mathcal{ALCHOIQ}^+$) KB is RSA (resp. RSA$^+$).

Theorem 2.2 ([14], Theorem 3)**.** *If $\mathcal{K}$ is RSA, then the size of $M[\mathcal{P}_{\mathcal{K}}]$ is polynomial in the size of $\mathcal{K}$.*

Tractability of standard reasoning tasks for RSA ontologies follows from Theorem 2.1 and Theorem 2.2.

[9] An *oriented forest* is a directed acyclic graph whose underlying undirected graph is a forest.

Table 2
Translation of Horn-$\mathcal{ALCHOIQ}$ axioms to build $E_{\mathcal{K}}$

Axioms in $\mathcal{K}$	LP rules
non-(T5) axiom α	$\pi(\alpha)$
$R \sqsubseteq S, * \in \{f, b\}$	$R^*(x, y) \rightarrow S^*(x, y)$
R role, $* \in \{f, b\}$	$R^*(x, y) \rightarrow R(x, y)$
	$R^f(x, y) \rightarrow Inv(R)^b(y, x)$
	$R^b(x, y) \rightarrow Inv(R)^f(y, x)$
(T5) axiom, R unsafe	$A(x) \rightarrow R^f(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$
(T5) axiom, R safe	$A(x) \wedge \texttt{notIn}(x, \texttt{unfold}(A, R, B)) \rightarrow R^f(x, v^{A,0}_{R,B}) \wedge B(v^{A,0}_{R,B})$
	$A(v^{A,i}_{R,B}) \rightarrow R^f(v^{A,i}_{R,B}, v^{A,i+1}_{R,B}) \wedge B(v^{A,i+1}_{R,B})$, if $R \in \texttt{confl}(R)$, for $i = 0, 1$
	$A(x) \rightarrow R^f(x, v^{A,1}_{R,B}) \wedge B(v^{A,1}_{R,B})$, for every $x \in \texttt{cycle}(A, R, B)$

2.3.1. Combined approach for RSA

The combined approach for RSA consists of two main steps to be offloaded to a Datalog reasoner able to handle *stratified negation* and *function symbols*.

The first step computes the canonical model of an RSA ontology over an extended signature (introduced to deal with *inverse roles* and *directionality* of newly generated binary atoms). The computed canonical model is not universal and, as such, might lead to spurious answers in the evaluation of CQs.

The second step of the computation performs a filtration of the computed answers to identify only the *certain answers* to the input query.

Canonical model computation The computation of the canonical model for a knowledge base $\mathcal{K}$ is performed by computing the LHM of the definite program obtained by translating $\mathcal{K}$ according to the rules in Table 2. The translation is an enhanced version of the translation given in Table 1 where axioms of type (T5) are *Skolemized* if the role involved is unsafe, and *constant Skolemized* otherwise. Constant Skolemization of some axioms can introduce *forks* in the canonical model that can lead to spurious answers. In order to keep track of these forks, directionality is taken into account when Skolemizing an axiom; roles are annotated with the direction in which they are "generated" (during the materialization process), and the annotation is propagated according to axioms in the RBox. This is still not enough to detect spurious forks in the canonical model and, in particular, cycles of length one (self-loops) or two can be the source of ambiguity during materialization. In order to solve the ambiguity of the canonical model, cycles of length one and two are unfolded into cycles of length three and four, respectively.

First we define the Datalog program $E_{\mathcal{K}}$ used to compute the canonical model for $\mathcal{K}$.

Definition 2.5 ([22], Definition 4)**.** Let $\texttt{confl}(R)$ be the set of roles S s.t. $R \sqsubseteq^*_{\mathcal{R}} T$ and $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$ for some T. Let $\prec$ be a strict total order on triples (A, R, B), with R safe and A, B concept names in $\mathcal{K}$. For each (A, R, B), let $v^{A,0}_{R,B}$, $v^{A,1}_{R,B}$ and $v^{A,2}_{R,B}$ be fresh constants; let $\texttt{self}(A, R, B)$ be the smallest set containing $v^{A,0}_{R,B}$ and $v^{A,1}_{R,B}$ if $R \in \texttt{confl}(R)$; and let $\texttt{cycle}(A, R, B)$ be the smallest set of terms containing, for each $S \in \texttt{confl}(R)$,

- $v^{D,0}_{S,C}$ if $(A, R, B) \prec (D, S, C)$;
- $v^{D,1}_{S,C}$ if $(D, S, C) \prec (A, R, B)$;
- $f^D_{S,C}(v^{D,0}_{S,C})$ and each $f^F_{T,E}(v^{D,0}_{S,C})$ s.t. $u^D_{S,C} \approx u^F_{T,E}$ is in M_{RSA}, if S is unsafe.

Finally, $\texttt{unfold}(A, R, B) = \texttt{self}(A, R, B) \cup \texttt{cycle}(A, R, B)$.

Let R^f and R^b be fresh binary predicates for each role R in $\mathcal{K}$, let $\texttt{NI}$ be a fresh unary predicate, and $\texttt{notIn}$ be a built-in predicate which holds when the first argument is *not* an element of the set given as the second element. Let $\mathcal{P}$ be the smallest program with a rule $\rightarrow \texttt{NI}(a)$ for each constant a and all rules in Table 2. We define $E_{\mathcal{K}} = \mathcal{P}^{\approx,\top}$.

The set $\texttt{confl}(R)$ intuitively contains the roles that are source of ambiguity in conjunction with R and hence need to be potentially unfolded if part of a loop; the arbitrary order $\prec$ determines the direction in which the loops are unfolded. Since the input ontology is RSA, there is no loop introduced by unsafe roles, and hence axiom of type (T5) involving unsafe roles don't need to be constant Skolemized.

Table 3
Rules in $\mathcal{P}_q$. Variables u, v, w from U are distinct

(1)	$\psi(\vec{x}, \vec{y}) \rightarrow \texttt{QM}(\vec{x}, \vec{y})$
(2)	$\rightarrow \texttt{named}(a)$ for each constant a in $\mathcal{O}$
(3a)	$\texttt{QM}(\vec{x}, \vec{y}), \mathit{not}\ \texttt{NI}(y_i) \rightarrow id(\vec{x}, \vec{y}, i, i)$ for each $1 \leqslant i \leqslant \lvert\vec{y}\rvert$
(3b)	$id(\vec{x}, \vec{y}, u, v) \rightarrow id(\vec{x}, \vec{y}, v, u)$
(3c)	$id(\vec{x}, \vec{y}, u, v), id(\vec{x}, \vec{y}, v, w) \rightarrow id(\vec{x}, \vec{y}, u, w)$
(4a)	for all $R(s, y_i), S(t, y_j)$ in q with $y_i, y_j \in \vec{y}$ $R^f(s, y_i) \wedge S^f(t, y_j) \wedge id(\vec{x}, \vec{y}, i, j) \wedge \mathit{not}\ s \approx t \rightarrow \texttt{fk}(\vec{x}, \vec{y})$
(4b)	for all $R(s, y_i), S(y_j, t)$ in q with $y_i, y_j \in \vec{y}$ $R^f(s, y_i) \wedge S^b(y_j, t) \wedge id(\vec{x}, \vec{y}, i, j) \wedge \mathit{not}\ s \approx t \rightarrow \texttt{fk}(\vec{x}, \vec{y})$
(4c)	for all $R(y_i, s), S(y_j, t)$ in q with $y_i, y_j \in \vec{y}$ $R^b(y_i, s) \wedge S^b(y_j, t) \wedge id(\vec{x}, \vec{y}, i, j) \wedge \mathit{not}\ s \approx t \rightarrow \texttt{fk}(\vec{x}, \vec{y})$
	for all $R(y_i, y_j), S(y_k, y_l)$ in q with $y_i, y_j, y_k, y_l \in \vec{y}$
(5a)	$R^f(y_i, y_j) \wedge S^f(y_k, y_l) \wedge id(\vec{x}, \vec{y}, j, l) \wedge y_i \approx y_k \wedge \mathit{not}\ \texttt{NI}(y_i) \rightarrow id(\vec{x}, \vec{y}, i, k)$
(5b)	$R^f(y_i, y_j) \wedge S^b(y_k, y_l) \wedge id(\vec{x}, \vec{y}, j, k) \wedge y_i \approx y_l \wedge \mathit{not}\ \texttt{NI}(y_i) \rightarrow id(\vec{x}, \vec{y}, i, l)$
(5c)	$R^b(y_i, y_j) \wedge S^b(y_k, y_l) \wedge id(\vec{x}, \vec{y}, i, k) \wedge y_j \approx y_l \wedge \mathit{not}\ \texttt{NI}(y_j) \rightarrow id(\vec{x}, \vec{y}, j, l)$
(6)	for each $R(y_i, y_j)$ in q with $y_i, y_j \in \vec{y}$ and $* \in \{f, b\}$ $R^*(y_i, y_j) \wedge id(\vec{x}, \vec{y}, i, v) \wedge id(\vec{x}, \vec{y}, j, w) \rightarrow \texttt{AQ}^*(\vec{x}, \vec{y}, v, w)$
	for each $* \in \{f, b\}$
(7a)	$\texttt{AQ}^*(\vec{x}, \vec{y}, u, v) \rightarrow \texttt{TQ}^*(\vec{x}, \vec{y}, u, v)$
(7b)	$\texttt{AQ}^*(\vec{x}, \vec{y}, u, v) \wedge \texttt{TQ}^*(\vec{x}, \vec{y}, v, w) \rightarrow \texttt{TQ}^*(\vec{x}, \vec{y}, u, w)$
(8a)	$\texttt{QM}(\vec{x}, \vec{y}) \wedge \mathit{not}\ \texttt{named}(x) \rightarrow \texttt{sp}(\vec{x}, \vec{y})$ for each $x \in \vec{x}$
(8b)	$\texttt{fk}(\vec{x}, \vec{y}) \rightarrow \texttt{sp}(\vec{x}, \vec{y})$
(8c)	$\texttt{TQ}^*(\vec{x}, \vec{y}, v, v) \rightarrow \texttt{sp}(\vec{x}, \vec{y})$ for each $* \in \{f, b\}$
(9)	$\texttt{QM}(\vec{x}, \vec{y}) \wedge \mathit{not}\ \texttt{sp}(\vec{x}, \vec{y}) \rightarrow \texttt{Ans}(\vec{x})$

The canonical model for an RSA input ontology is defined as $M[E_{\mathcal{K}}]$.

Theorem 2.3 ([22],Theorem 3)**.** *The following holds:*

(i) $M[E_{\mathcal{K}}]$ *is polynomial in* $|\mathcal{K}|$*;*
(ii) $\mathcal{K}$ *is satisfiable iff* $E_{\mathcal{K}} \not\models \exists y.\bot(y)$*;*
(iii) *if* $\mathcal{K}$ *is satisfiable,* $\mathcal{K} \models A(c)$ *iff* $A(c) \in M[E_{\mathcal{K}}]$*;*
(iv) *there are no terms* s, t *and role* R *s.t.* $E_{\mathcal{K}} \models R^f(s, t) \wedge R^b(s, t)$.

Filtering spurious answers For the filtering step, a *query dependent* logic program $\mathcal{P}_q$ is introduced to filter out all spurious answers to an input query q over the extended canonical model $M[E_{\mathcal{K}}]$ computed in the previous section. The program identifies and discards any match that cannot be enforced by a TBox alone and hence correspond to spurious answers introduced by the canonical model. This includes the task of detecting forks and cycles in the model and answers that contain *anonymous terms* (i.e., functional terms and constants introduced as part of the canonical model program). Rules for the filtering program are provided in Table 3.

Filtering program $\mathcal{P}_q$ and its extension $\mathcal{P}_{q,\mathcal{K}}$ with $E_{\mathcal{K}}$ from Def. 2.5 are defined as follows.

Definition 2.6 ([22], Definition 5)**.** Let $q = \exists \vec{y}.\psi(\vec{x}, \vec{y})$ be a CQ, let QM, sp, and fk be fresh predicates of arity $|\vec{x}| + |\vec{y}|$, let id, $\texttt{AQ}^*$, $\texttt{TQ}^*$ with $* \in \{f, b\}$ be fresh predicates of arity $|\vec{x}| + |\vec{y}| + 2$, let Ans be a fresh predicate of arity $|\vec{x}|$, let named be a fresh unary predicate, and let U be a set of fresh variables s.t. $|U| \geqslant |\vec{y}|$. Then, $\mathcal{P}_q$ is the smallest program with all rules in Table 3, and $\mathcal{P}_{q,\mathcal{K}}$ is defined as $E_{\mathcal{K}} \cup \mathcal{P}_q$.

Theorem 2.4 ([22], Theorem 4)**.** *Let* $\mathcal{P}_q$ *be the filtering program for* q*, and* $\mathcal{P}_{q,\mathcal{K}} = E_{\mathcal{K}} \cup \mathcal{P}_q$*. It holds that [22]:*

(i) $\mathcal{P}_{q,\mathcal{K}}$ *is* stratified*;*

(ii) $M[\mathcal{P}_{q,\mathcal{K}}]$ *is polynomial in* $|\mathcal{K}|$ *and exponential in* $|q|$*;*
(iii) *if* $\mathcal{K}$ *is satisfiable,* $\vec{x} \in cert(q, \mathcal{K})$ *iff* $\mathcal{P}_{q,\mathcal{K}} \models \texttt{Ans}(\vec{x})$.

We can then build a worst-case exponential algorithm that, given an ontology $\mathcal{K}$ and a CQ q, it materializes $\mathcal{P}_{q,\mathcal{K}}$ and returns all instances of predicate `Ans`. We obtain a "guess and check" algorithm that leads to an NP–completeness result for BCQs [22]. The algorithm first materializes $E_{\mathcal{K}}$ in polynomial time and then guesses a match σ to q over the materialization; finally it computes $(\mathcal{P}_{q,\mathcal{K}})\sigma$.

Theorem 2.5 ([22], Theorem 5)**.** *Checking whether* $\mathcal{K} \models q(\vec{x}, \vec{y})$ *with* $\mathcal{K}$ *an RSA ontology and* $q(\vec{x}, \vec{y})$ *a BCQ is* NP*–complete in combined complexity.*[10]

3. Overview

We propose a hybrid query answering architecture which provides CQ answering capabilities for OWL 2 by means of combining different answering services treated as black-boxes. In particular, we combine scalable CQ answering services targeting tractable ontology languages with answering services for more expressive languages approaching the full OWL 2.

Given an input CQ over a certain knowledge base, we process the query using the tractable services; if the query can be fully answered by one of these tractable services, we simply provide the resulting answers to the user. Otherwise, we compute multiple lower and upper bounds to the answers to the query by approximating the knowledge base "from above" and "from below" and taking the union of the lower bounds and the intersection of the upper bounds. Finally, if the bounds do not coincide, the "gap" answers are validated by using the "full" service.

As part of this work, we introduce a novel algorithm to compute a lower bound to the answers to an input query by means of approximation to RSA. Similarly, we propose an algorithm to compute an approximation "from above" targetting RSA$^+$.

The reference implementation ACQuA is built on top of the following tools:

(i) RSAComb, a novel system for CQ answering over RSA ontologies, based on the combined approach, extended with algorithms to computes bounds of the answers to a query via approximation of the input KB to RSA;
(ii) PAGOdA, providing lower and upper bounds to the answers to a query and techniques to further refine these bounds to provide CQ answering capability over OWL 2 DL;
(iii) a fully-fledged reasoner (such as HermiT) for CQ answering over a certain ontology language.

These tools allowed us to build a fine-grained "pay-as-you-go" approach, offering suitable, performant solutions depending on the inputs to the system; overall, this results in a lower complexity of the answer computation, when support for high expressivity is not needed. Note that we included both RSAComb and PAGOdA in ACQuA because, in general, the bounds computed by RSAComb are *incomparable* with the ones produced by PAGOdA, as we will show in Sections 4 and 5. However, any of these components could be potentially substituted or augmented with more capable ones; in particular, any relevant service mentioned above could be used in ACQuA (e.g., the fully-fledged reasoner HermiT could be substituted with Konclude).

Given a generic KB $\mathcal{K} = \langle \mathcal{T} \cup \mathcal{R}, \mathcal{A} \rangle$ and a CQ $q(\vec{x}) = \exists \vec{y} \varphi(\vec{x}, \vec{y})$ containing only symbols in $\mathcal{K}$, the combination of RSAComb, PAGOdA, and HermiT performs the following steps to compute the full set of answers to $q(\vec{x})$ over $\mathcal{K}$ (see Fig. 2).

1. A preliminary satisfiability check is performed over the input knowledge base $\mathcal{K}$. The procedure terminates if $\mathcal{K}$ is unsatisfiable.
2. If $\mathcal{K}$ is either $\mathcal{RL}$ or $\mathcal{ELHO}^r_\bot$ return the answers provided by the lower bound algorithm in PAGOdA.[11] Otherwise, proceed to step 3.
3. If $\mathcal{K}$ is RSA, return the full set of answers computed by RSAComb.[12] Otherwise, proceed to step 4.

[10] It is worth noting that, to the best of our knowledge, at the moment a similar bound for RSA$^+$ is not know.
[11] In this case, $\mathcal{K}$ falls in one of the profiles for which the lower bound computation in PAGOdA is sound and complete for CQ answering.
[12] In this case, RSAComb provides a sound and complete algorithm for CQ answering over $\mathcal{K}$.

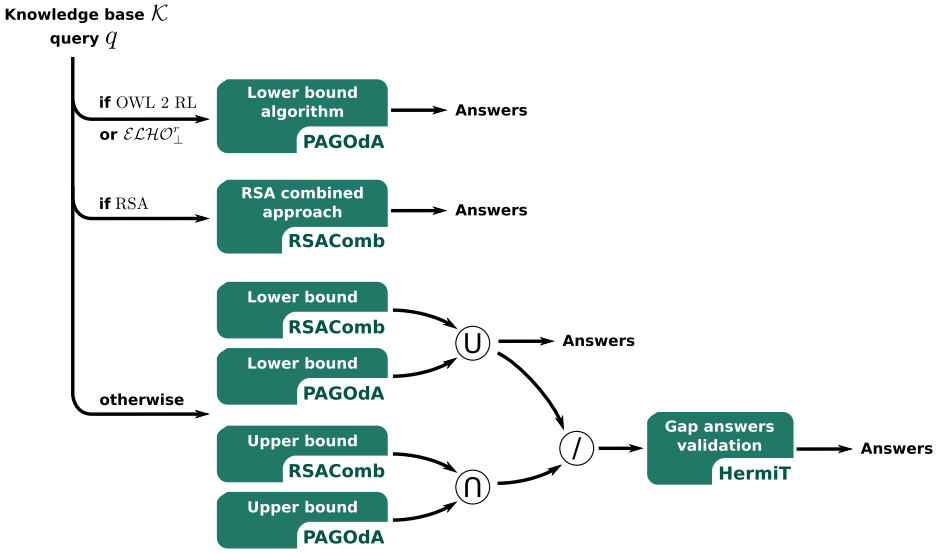


Fig. 2. Workflow of the ACQuA system.

4. Compute the bounds to the answers to q as $L^q = L^q_P \cup L^q_R$ and $U^q = U^q_P \cap U^q_R$, with $\langle L^q_P, U^q_P \rangle$ and $\langle L^q_R, U^q_R \rangle$ the lower and upper bounds computed by PAGOdA and RSAComb, respectively.
5. If $G^q = U^q \setminus L^q = 0$, return L^q. Otherwise, proceed to step 6.
6. Compute $\mathcal{K}^q$, a subset of $\mathcal{K}$, relevant for the answering of $q(\vec{x})$.
7. Use HermiT on $\mathcal{K}^q$, to check the entailment of the answers in G^q and remove any remaining spurious answer.
8. Return $L^q \cup G^q$.

The choice of fully-fledged reasoner ultimately determines the class of ontologies for which CQ answering is sound and complete under ground and/or certain answer semantics for the overall system. Thanks to RSAComb, ACQuA is sound and complete for CQ answering under certain answer semantics for ontologies in RSA [22]. With the integration of PAGOdA, and a suitable fully-fledged reasoner, like HermiT, ACQuA is able to answer internalisable queries [36] over OWL 2 DL under certain answer semantics [85].

Steps 2,6,7 and the computation of L^q_P, U^q_P in step 4 are offloaded to PAGOdA; we refer the reader to [85] for more details. We will instead focus our attention on the underlying RSAComb reasoner; in particular we dedicate Sections 4–5 to the description of the novel algorithms used in step 4 to compute L^q_R, U^q_R via approximation to RSA. In Section 6 we provide more details on the design and architecture of ACQuA and RSAComb (both as a standalone system and its integration in ACQuA).

To help the reader follow along with the description of the proposed techniques, we consider the following running example.

Example 3.1. Consider the KB $\mathcal{K}_{ex} = \langle \mathcal{T}_{ex} \cup \mathcal{R}_{ex}, \mathcal{A}_{ex} \rangle$ and the CQs $\mathcal{Q}_{ex}$, with ABox $\mathcal{A}_{ex}$ containing assertions (a1)–(a10), TBox $\mathcal{T}_{ex}$ containing axioms (t1)–(t9), RBox $\mathcal{R}_{ex}$ containing axioms (r1)–(r2), and $\mathcal{Q}_{ex}$ containing queries (q1)–(q2) in Table 4.

Intuitively, the ABox contains a collection of statements about researchers and their research outputs; on top of that, the ontology (RBox and TBox) models additional information about the relationships between different types of papers and their publications processes. The CQs ask about venues and works published in multiple venues. For reader's convenience, a visual representation of the ABox $\mathcal{A}_{ex}$ is shown in Fig. 3.

Some axioms are not expressed in normal form (see Table 1) and can be further normalized as follows: axiom (t1) can be rewritten as

$$\texttt{PhDStudent} \sqsubseteq \texttt{Student} \qquad \texttt{PhDStudent} \sqsubseteq \texttt{Researcher} \tag{t1bis}$$

Table 4
Running example $\mathcal{K}_{ex}$

(r1)	$\texttt{published} \equiv \texttt{publishedBy}^-$	(a1)	$\texttt{PhDStudent}(bart)$
(r2)	$\texttt{accepts} \sqsubseteq \texttt{reviews}$	(a2)	$\texttt{Researcher}(lisa)$
		(a3)	$\texttt{Journal}(journal1)$
(t1)	$\texttt{PhDStudent} \sqsubseteq \texttt{Student} \sqcap \texttt{Researcher}$	(a4)	$\texttt{Journal}(journal2)$
(t2)	$\texttt{JournalPaper} \sqcap \texttt{Thesis} \sqsubseteq \bot$	(a5)	$\texttt{Journal}(journal3)$
(t3)	$\texttt{Report} \sqsubseteq \texttt{Paper} \sqcup \texttt{Thesis}$	(a6)	$\texttt{writes}(bart, work1)$
(t4)	$\texttt{Journal} \sqsubseteq \exists \texttt{published}.\texttt{Paper}$	(a7)	$\texttt{writes}(lisa, work1)$
(t5)	$\texttt{Researcher} \sqsubseteq \exists \texttt{writes}.\texttt{Paper}$	(a8)	$\texttt{published}(journal1, work1)$
(t6)	$\texttt{Paper} \sqsubseteq \exists \texttt{presentedAt}.\texttt{Conference}$	(a9)	$\texttt{JournalPaper}(work1)$
(t7)	$\texttt{Paper} \sqsubseteq \leqslant 1\, \texttt{presentedAt}.\texttt{Conference}$	(a10)	$\texttt{Report}(work1)$
(t8)	$\texttt{Conference} \sqsubseteq \exists \texttt{accepts}.\texttt{Paper}$		
(t9)	$\exists \texttt{reviews}^-.\texttt{Conference} \sqsubseteq \texttt{ConferencePaper}$		
(q1)	$q_1(x_2) = \texttt{publishedBy}(x_1, x_2)$		
(q2)	$q_2(x_1, x_2) = \texttt{published}(x_1, x_3) \wedge \texttt{published}(x_2, x_3) \wedge x_1 \neq x_2$		

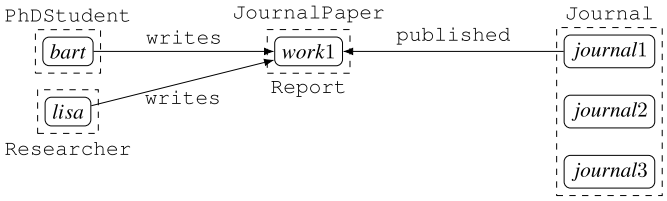


Fig. 3. Graphical representation of the ABox $\mathcal{A}_{ex}$.

while axiom (r1) becomes

$$\texttt{published} \sqsubseteq \texttt{publishedBy}^- \qquad \texttt{publishedBy} \sqsubseteq \texttt{published}^- \tag{r1bis}$$

Note that $\mathcal{K}_{ex}$ is not in Horn-$\mathcal{ALCHOIQ}$ because of axiom (t3), and hence it is neither RSA nor RSA$^+$.

4. Lower bound computation

In this section we present a technique to compute a lower bound to the answers to an input query, by means of approximating the input KB to RSA.

RSA is not purely syntactically defined, and instead introduces a set of constraints over the ontology language Horn-$\mathcal{ALCHOIQ}$; as such, the naïve approximation that consists in discarding any axiom type which is not in the target approximation language does not work. Instead, we split our approximation algorithm in three sub-steps, each building on top of the previous one:

1. From a generic $\mathcal{SROIQ}$ ontology to $\mathcal{ALCHOIQ}$ by discarding any axiom that is not in the target language;
2. From $\mathcal{ALCHOIQ}$ to Horn-$\mathcal{ALCHOIQ}$ by means of *program shifting*;
3. From Horn-$\mathcal{ALCHOIQ}$ to RSA by modifying the input KB in order to enforce the constraints imposed by the RSA definition.

4.1. Approximation to $\mathcal{ALCHOIQ}$

This first step is performed by discarding any axiom that is not in $\mathcal{ALCHOIQ}$, namely axioms of type (R3)–(R4) and (T6)–(T7) in Table 1.

Let $\mathcal{K}'$ be the $\mathcal{ALCHOIQ}$ restriction of a KB $\mathcal{K}$. By the monotonicity of FOL all certain answers w.r.t. $\mathcal{K}'$ are also certain answers w.r.t. $\mathcal{K}$; in other words, discarding axioms in a KB will lead to having more models, which, in turn, leads to fewer answers, under certain answer semantics. Moreover, if $\mathcal{K}'$ is unsatisfiable, so is $\mathcal{K}$.

In Example 3.1, $\mathcal{K}_{ex}$ is in $\mathcal{ALCHOIQ}$, so no axioms are discarded.

4.2. Approximation to Horn-$\mathcal{ALCHOIQ}$

We will now describe how to reduce an $\mathcal{ALCHOIQ}$ ontology to Horn-$\mathcal{ALCHOIQ}$. This involves the approximation of axioms of type (T1), (T4) by eliminating the disjunction in the head of the axioms.[13] Simply discarding them is not desirable, especially when considering that *disjunctive axioms* are quite common in practice.

To address this and improve the approximation to Horn-$\mathcal{ALCHOIQ}$ we rely on a technique known as *program shifting* [18] to convert disjunctive Datalog rules into Datalog. Program shifting is a polynomial compilation of disjunctive logic rules into Datalog rules that preserve soundness of CQ answering, and acts on the translation $\pi(\cdot)$ of the axioms into definite rules.

Example 4.1. In Example 3.1, we know that assertions $\texttt{Report}(work1)$ and $\texttt{JournalPaper}(work1)$ hold (because of assertions (a10), (a9)). Moreover, by (t2), we know that $\texttt{Thesis}(work1)$ does *not* hold. Using this information, along with (t3), we can derive $\texttt{Paper}(work1)$. This derivation is deterministic and can be captured by Datalog rules. To make this reasoning explicit, we introduce a fresh atom $\overline{\texttt{Thesis}}$ that intuitively represents the *complement* of $\texttt{Thesis}$, and add the following axioms to $\mathcal{K}_{ex}$:

$$\texttt{JournalPaper} \sqsubseteq \overline{\texttt{Thesis}} \qquad \texttt{Report} \sqcap \overline{\texttt{Thesis}} \sqsubseteq \texttt{Paper} \tag{8}$$

These rules can be used to derive $\texttt{Paper}(work1)$.

Program shifting is formally defined as follows.

Definition 4.1 ([85], Def. 4.3)**.** Let r be a normalized disjunctive Datalog rule. For each predicate P in r, let $\overline{P}$ be a fresh predicate of the same arity. The *shifting* of r, denoted $\texttt{shift}(r)$, is the following set of rules:

– if r is of the form

$$\beta_1 \wedge \cdots \wedge \beta_n \rightarrow \bot \tag{9}$$

then

$$\texttt{shift}(r) = \{r\} \cup \{\beta_1 \wedge \cdots \wedge \beta_{i-1} \wedge \beta_{i+1} \wedge \cdots \wedge \beta_n \rightarrow \bar{\beta}_i \mid 1 \leqslant i \leqslant n\} \tag{10}$$

– if r is of the form

$$\beta_1 \wedge \cdots \wedge \beta_n \rightarrow \gamma_1 \vee \cdots \vee \gamma_m \tag{11}$$

then $\texttt{shift}(r)$ consists of the following rules:

$$\beta_1 \wedge \cdots \wedge \beta_n \wedge \bar{\gamma}_1 \wedge \cdots \wedge \bar{\gamma}_m \rightarrow \bot \tag{12}$$

$$\beta_1 \wedge \cdots \wedge \beta_i \wedge \bar{\gamma}_1 \wedge \cdots \wedge \bar{\gamma}_{j-1} \wedge \bar{\gamma}_{j+1} \wedge \ldots \bar{\gamma}_m \rightarrow \gamma_j \quad \text{for } 1 \leqslant j \leqslant m \tag{13}$$

$$\beta_1 \wedge \cdots \wedge \beta_{i-1} \wedge \beta_{i+1} \wedge \cdots \wedge \beta_n \wedge \bar{\gamma}_1 \wedge \ldots \bar{\gamma}_m \rightarrow \bar{\beta}_i \quad \text{for } 1 \leqslant i \leqslant n \tag{14}$$

[13] While axioms of type (T4) do not use disjunction explicitly, their translation into definite rules involve disjunction in the head of the rule.

This can be generalized to sets of rules Σ as follows:

$$\texttt{shift}(\Sigma) = \bigcup_{r \in \Sigma} \texttt{shift}(r) \tag{15}$$

We apply this technique to our $\mathcal{ALCHOIQ}$ KB in order to reduce ourselves to a Horn KB. This procedure guarantees to produce a *polynomial* approximation of the input KB which is sound (but not necessarily complete) w.r.t. CQ answering. For r a disjunctive Datalog rule with n atoms in the body and m atoms in the head, $\texttt{shift}(r)$ contains $n + m + 1$ rules.

Theorem 4.1. *Let $\mathcal{K}' = \langle \mathcal{O}', \mathcal{A} \rangle$ be the $\mathcal{ALCHOIQ}$ restriction of the KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$. Moreover, let $\mathcal{K}'' = \langle \texttt{shift}(\mathcal{O}'), \mathcal{A} \rangle$. Then $\texttt{cert}(q, \mathcal{K}'') \subseteq \texttt{cert}(q, \mathcal{K}')$.*

Proof. See the Appendix. □

4.3. Approximation to RSA

In this section we provide a description of an algorithm to approximate the Horn-$\mathcal{ALCHOIQ}$ KB $\mathcal{K}$ obtained in the previous step into an RSA KB $\mathcal{K}'$ such that $cert(q, \mathcal{K}') \subseteq cert(q, \mathcal{K})$ for any KB q. Given a Horn-$\mathcal{ALCHOIQ}$ KB $\mathcal{K}$, checking if $\mathcal{K}$ is RSA consists of the following steps (see Def. 2.3):

1. checking whether $G_{\mathcal{K}}$ is an *oriented forest*;
2. checking whether $\mathcal{K}$ is *equality safe*.

We first consider step 1. If $G_{\mathcal{K}}$ is not an oriented forest, then its underlying *undirected graph* has a cycle. In order to make $G_{\mathcal{K}}$ an *oriented forest* we want to detect these cycles, break them and propagate the changes back to $\mathcal{K}$.

Cycles can be broken by removing nodes from $G_{\mathcal{K}}$. Nodes in $G_{\mathcal{K}}$ are of the form $u^A_{R,B}$, paired with a corresponding existential axiom $A \sqsubseteq \exists R.B \in \mathcal{K}$ of type (T5). The action of deleting a node from the graph can be propagated back to $\mathcal{K}$ by removing the corresponding (T5) axiom. Due to monotonicity of FOL, deleting axioms from $\mathcal{K}$ produces a lower bound approximation of $\mathcal{K}$ w.r.t. CQ answering.

Lemma 4.1. *Let $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ be a Horn-$\mathcal{ALCHOIQ}$ KB, $G_{\mathcal{K}}$ be its dependency graph as defined in Def. 2.3 and $u^A_{R,B}$ a node in $G_{\mathcal{K}}$. The dependency graph $G_{\mathcal{K}'}$ corresponding to $\mathcal{K}' = \langle \mathcal{O} \setminus \{A \sqsubseteq \exists R.B\}, \mathcal{A} \rangle$ does not contain $u^A_{R,B}$.*

Proof. This is proven by observing that, by definition of dependency graph, constant $u^A_{R,B}$ can solely be introduced by the corresponding axiom $A \sqsubseteq \exists R.B$, and hence, removing the axiom from $\mathcal{O}$ will remove the node from $G_{\mathcal{K}'}$. □

Using the Datalog reasoner, we compute M_{RSA} from the program $\mathcal{P}^{\approx,\top}_{\text{RSA}}$ obtained from $\mathcal{K}$, and retrieve all instances of role $\texttt{E}$ to build $G_{\mathcal{K}}$. Finally, we use the modified DFS visit of the graph shown in Algorithm 1 to detect any cycle in $G_{\mathcal{K}}$. During the visit, the algorithm determines a representative node for each cycle, which will be the node selected to be removed. In order to keep the visit as efficient as possible we determine these nodes eagerly, by selecting the last processed node when a cycle is detected. Let D be this set of nodes, then for every $u^A_{R,B} \in D$ we remove the corresponding axiom $A \sqsubseteq \exists R.B$ in $\mathcal{K}$. Note that D is, in general, not unique and different such sets might lead to different lower bounds.

Next, we need to deal with *equality safety* (step 2). According to the definition of RSA, the following steps can be performed to ensure this property:

i. delete any (T4) axiom that involves a role S such that there exists $w \approx t$ (with w and t distinct) and $R(t, u^A_{R,B})$ in M_{RSA} and $R \sqsubseteq Inv(S)$;
ii. if there is a pair of atoms $R(a, u^A_{R,B})$, $S(u^A_{R,B}, a)$ in M_{RSA} with $a \in N_I$ and a role T such that both $R \sqsubseteq^*_{\mathcal{R}} T$ and $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$ hold, then remove some axiom of type (R2) to break the derivation chain that deduces either $R \sqsubseteq^*_{\mathcal{R}} T$ or $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$.

Input: Dependency graph $G_{\mathcal{K}}$ for KB $\mathcal{K}$
Output: Set of nodes C, representatives of each cycle in $G_{\mathcal{K}}$

```
1  let N be the set of nodes in G_K;
2  let C be an empty set;
3  foreach node n in N do
4      if n is not discovered then
5          let S be an empty stack;
6          push n in S;
7          while S is not empty do
8              pop v from S;
9              if v is not discovered then
10                 label v as discovered;
11                 let adj be the set of nodes adjacent to v;
12                 if any node in adj is discovered then
13                     push v in C;
14                 else
15                     foreach node w in adj do
16                         push w in S;
17 return C
```

Algorithm 1: Cycle detection in $G_{\mathcal{K}}$

Again, by removing some selected axioms we are able to force the input Horn-$\mathcal{ALCHOIQ}$ ontology to satisfy RSA additional constraints. In the following, we summarize steps 1–2 described above with the function $\texttt{lower}(\cdot)$ from KBs to KBs.

Theorem 4.2. *Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ KB, and $\mathcal{K}'$ its syntactic restriction to $\mathcal{ALCHOIQ}$. Then, we have that $cert(q, \texttt{lower}(\texttt{shift}(\mathcal{K}'))) \subseteq cert(q, \mathcal{K})$.*

Proof. By Section 4.1 and Theorem 4.1 we know that

$$\texttt{cert}\big(q, \texttt{shift}\big(\mathcal{K}'\big)\big) \subseteq \texttt{cert}\big(q, \mathcal{K}'\big) \subseteq \texttt{cert}(q, \mathcal{K}) \tag{16}$$

Moreover, we can observe that $\texttt{lower}(\cdot)$ only removes axioms from the input ontology; by monotonicity of FOL we have that $\texttt{cert}(q, \texttt{lower}(\texttt{shift}(\mathcal{K}'))) \subseteq \texttt{cert}(q, \texttt{shift}(\mathcal{K}')) \subseteq \texttt{cert}(q, \mathcal{K})$. □

As mentioned in Section 2, PAGOdA uses a similar approach to compute a lower bound by approximating the input ontology first to *disjunctive* Datalog and then to Datalog; this is done by discarding any axiom that is not in the language, while introducing some additional heuristics to handle specifically *disjunctive* and *existential* axioms.

Note that, in general, the lower bound resulting from the algorithm proposed here is *incomparable* with the one produced by PAGOdA.

The next example shows a scenario in which the lower bound computed by our algorithm is tighter than the one produced by PAGOdA. On the other hand, the RSA language fully captures OWL 2 RL (used internally by PAGOdA) only when not considering *property chain axioms*.

Both approximation techniques will be later combined into ACQuA.

Example 4.2. Consider our running Example 3.1 and $\mathcal{K}'_{ex} = \texttt{shift}(\mathcal{K}_{ex})$. Then

- `presentedAt` is unsafe because of axioms (t6), (t7);
- `accepts` is unsafe because of axioms (t8), (t9) and (r2);

whereas all other roles are safe.

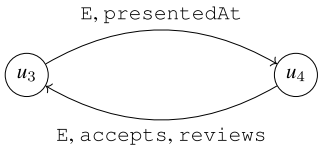


Fig. 4. Graphical representation of $G_{\mathcal{K}_{ex}}$.

Now, let u_i, for $1 \leqslant i \leqslant 4$ be unique, fresh constants, and $\mathcal{P}^{ex}_{\text{RSA}}$ be the logic program (according to Def. 2.3), corresponding to $\mathcal{K}'_{ex}$. In particular

$$\texttt{Journal}(x) \rightarrow \texttt{published}(x, u_1) \wedge \texttt{PE}(x, u_1) \wedge \texttt{Paper}(u_1) \tag{17}$$

$$\texttt{Researcher}(x) \rightarrow \texttt{writes}(x, u_2) \wedge \texttt{PE}(x, u_2) \wedge \texttt{Paper}(u_2) \tag{18}$$

$$\texttt{Paper}(x) \rightarrow \texttt{presentedAt}(x, u_3) \wedge \texttt{PE}(x, u_3) \wedge \texttt{Conference}(u_3) \wedge \texttt{U}(u_3) \tag{19}$$

$$\texttt{Conference}(x) \rightarrow \texttt{accepted}(x, u_4) \wedge \texttt{PE}(x, u_4) \wedge \texttt{Paper}(u_4) \wedge \texttt{U}(u_4) \tag{20}$$

is the translation (according to Def. 2.3) of (t4), (t5), (t6), and (t8). Finally, M^{ex}_{RSA} is the LHM of $\mathcal{P}^{ex}_{\text{RSA}}$.

The dependency graph $G_{\mathcal{K}'_{ex}}$ is shown in Fig. 4. $G_{\mathcal{K}'_{ex}}$ is not an oriented forest and as such we need to detect a set of nodes whose removal will turn the graph in Fig. 4 into a tree. Let's assume Algorithm 1 returns the set $\{u_4\}$ to be removed from $G_{\mathcal{K}'_{ex}}$. We propagate this change to $\mathcal{K}'_{ex}$ by removing axiom (t8). $\mathcal{K}'_{ex}$ was already equality safe. We denote the KB resulting from this process with $\mathcal{K}''_{ex} = \texttt{lower}(\mathcal{K}'_{ex})$.

If we consider the query (q1), then:

$$\texttt{cert}\big(q_1, \mathcal{K}''_{ex}\big) = \{\textit{journal}1, \textit{journal}2, \textit{journal}3\}. \tag{21}$$

It can be verified that the lower bound computed by PAGOdA is not as tight and results in the set of answers containing only *journal*1.

5. Upper bound computation

We will now look at the problem of approximating a generic input KB $\mathcal{K}$ to a KB $\mathcal{K}'$ *from above*, such that answering an input query over the approximated KB will return an upper bound to the answers. More formally, given an input KB $\mathcal{K}$, we want to find a KB $\mathcal{K}'$ s.t. $\mathit{cert}(q, \mathcal{K}) \subseteq \mathit{cert}(q, \mathcal{K}')$ for any CQ q. We initially consider $\mathcal{ALCHOIQ}^+$ as the source ontology language, not taking property chain axioms (T4) into account, and approximate the ontology to RSA$^+$. Some additional comments on how to handle axioms (T4) will also be provided.

We adopt a similar approach to the one used in the lower bound computation and divide the procedure in steps. Given an $\mathcal{ALCHOIQ}^+$ KB, we proceed as follows

1. replace any occurrence of $\bot$ in the knowledge base with a fresh nullary predicate $\bot_f$ with no special meaning;
2. approximate disjunctive rules by removing all but one disjunct from the head of the rule. For each rule, the selected disjunct is chosen deterministically using an efficient *choice function*;
3. enforce the constraints that define the RSA ontology language on the Horn-$\mathcal{ALCHOIQ}^+$ KB obtained in the previous step.

5.1. $\bot$ substitution

As described above, ACQuA performs a preliminary satisfiability check on the input KB; in spite of this, while strengthening the KB, we might cause the KB to become unsatisfiable.

In order to provide a meaningful upper bound even in cases where the approximation leads to an unsatisfiable KB, we adopt an approach initially proposed in PAGOdA. The idea is to substitute every occurrence of $\bot$ with a

fresh nullary predicate $\bot_f$ with no predefined meaning; by doing so we avoid the derivation of the entire Herbrand base, ignoring the fact that the final KB approximation might be unsatisfiable. Note that, despite the fact that $\bot$ is stripped of its built-in semantics in FOL, weakening the KB, it can be shown (see [85, Lemma 5.4,Theorem 5.5]) that we can still compute a meaningful upper bound for any input query.

This step has been included purely for theoretical purposes. RDFox, used in the implementation of the approximation algorithm, will not explicitly check for satisfiability during query answering, making it possible to consider correct the answers to a query even when the KB is unsatisfiable.

5.2. Approximation of disjunctive rules

According to Table 1, axioms of type (T1) and (T4) can introduce disjunction in the head of rules. This usually results in non-determinism in the answering process and a corresponding jump in computational complexity. In order to rewrite these axioms and avoid the introduction of this operator, we borrow a technique used in a similar fashion in PAGOdA. The approach consists in replacing any disjunction in the head of a rule with one of the disjuncts. It is easy to see that this strengthens the KB and eliminates any non-determinism introduced by the disjunction. The surviving disjunct is chosen deterministically using an efficient choice function; the idea is to analyse the dependency graph of the KB and choose a disjunct which does not eventually lead to a contradiction. To this end, a standard dependency graph of the KB is built and disjuncts are ordered according to their distance from $\bot_f$; The ordering is used to select those disjuncts that are less likely to lead to a contradiction, by picking (one of) the furthest from $\bot_f$. For more details on the definition of a choice function, see [85, Section 8.2].

Given a choice function $\mathtt{ch}$ that returns a concept name out of an input set, we define the process of eliminating disjunction from the head of a rule as follows

Definition 5.1. Let δ be a function from axioms to axioms eliminating disjunction from the head of any axiom of type (T1) and (T4):

$$\delta(\alpha) = \begin{cases} \bigsqcap_{i=1}^{n} A_i \sqsubseteq \mathtt{ch}(\{B_j \mid 1 \leqslant j \leqslant m\}) & \text{if } \alpha \equiv \bigsqcap_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{j=1}^{m} B_j \\ A \sqsubseteq \leqslant 1 R.B & \text{if } \alpha \equiv A \sqsubseteq \leqslant m R.B \\ \alpha & \text{otherwise} \end{cases} \tag{22}$$

The definition of δ can be trivially extended to sets of axioms and KBs.

Example 5.1. Consider axioms (t3) from our running example. Let $\mathtt{ch}$ be a choice function, such that

$$\mathtt{ch}\big(\{\mathtt{Paper}, \mathtt{Thesis}\}\big) = \mathtt{Paper} \tag{23}$$

Then δ(t3) is

$$\mathtt{Report} \sqsubseteq \mathtt{Paper} \tag{t3'}$$

5.3. From Horn-$\mathcal{ALCHOIQ}^+$ to RSA$^+$

The final step of the approximation process consists in enforcing the additional constraints that the RSA language introduces on top of Horn-$\mathcal{ALCHOIQ}$. We apply these constraints on top of the KB obtained in the previous step, which is Horn-$\mathcal{ALCHOIQ}^+$, obtaining an RSA$^+$ KB. We will later prove that the algorithm for the combined approach for RSA applied to an RSA$^+$ ontology is complete w.r.t. CQ answering.

Given $\mathcal{K}$ a Horn-$\mathcal{ALCHOIQ}^+$ KB and $G_{\mathcal{K}}$ its dependency graph as defined in Def. 2.3, checking if $\mathcal{K}$ is RSA$^+$ consists of:

1. checking whether $G_{\mathcal{K}}$ is an *oriented forest*;
2. checking whether $\mathcal{K}$ is *equality safe*.

In order to ensure equality safety we proceed similarly to the lower bound case. For any pair of atoms $w \approx t$, $R(t, u^A_{R,B}) \in M_{RSA}$ and role S s.t. $R \sqsubseteq Inv(S)$, if S occurs in an axiom $\alpha \equiv C \sqsubseteq \leqslant 1S.D$ of type (T4), we convert α into the axiom $C \sqcap \exists S.D \sqsubseteq \bot$. It is easy to see that, for any C, $D \in N_C$ and S role, $\{C \sqcap \exists S.D \sqsubseteq \bot\} \models C \sqsubseteq \leqslant 1S.D$ and hence the rewriting is a strengthening of the KB.

On the other hand, for each pair of atoms $R(a, u^A_{R,B})$, $S(u^A_{R,B}, a) \in M_{RSA}$, with $a \in N_I$ and role T such that $R \sqsubseteq^*_{\mathcal{R}} T$ and $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$, we know that term $u^A_{R,B}$ was introduced by an axiom $A \sqsubseteq \exists R.B$ of type (T5). In order to satisfy the constraint, we *mark* this axiom for constant Skolemization, meaning that when translated into a logic rule this axiom will be translated into $A(x) \rightarrow R(x, c) \wedge B(c)$ for some unique fresh constant c.[14] Moreover, we assume to have a boolean function $\texttt{marked}(\alpha)$ over axioms that returns *true* if α is a marked axiom.

Finally, we reduce $G_{\mathcal{K}}$ to an oriented forest. We proceed similarly to the lower bound computation described in Section 4.3. In fact, we can reuse Algorithm 1 to gather a possible set of nodes D, whose removal would render the dependency graph an oriented forest. As explained before, each node $u^A_{R,B}$ uniquely identifies an axiom $A \sqsubseteq \exists R.B$ of type (T5) in the input KB. In order to break the cycles while strengthening the KB we mark the axioms in D for constant Skolemization.

These steps can be summarized in the definition of δ':

Definition 5.2. We define δ' as a function from axioms to sets of axioms.

$$\delta'(\alpha) = \begin{cases} \{C \sqcap \exists S.D \sqsubseteq \bot_f\} & \text{if } \alpha \equiv C \sqsubseteq \leqslant 1S.D \text{ and } \exists R \text{ unsafe s.t. } R \sqsubseteq Inv(S) \text{ and} \\ & w \approx t,\ R(t, u^A_{R,B}) \in M_{RSA} \text{ with } w, t \text{ distinct} \\ \{A \sqsubseteq \exists R.\{b^A_{R,B}\}, \{b^A_{R,B}\} \sqsubseteq B\} & \text{if } \alpha \equiv A \sqsubseteq \exists R.B \text{ and } \texttt{marked}(\alpha) \\ \{\alpha\} & \text{otherwise} \end{cases} \tag{24}$$

where $b^A_{R,B}$ is a fresh constant, unique to axiom $A \sqsubseteq \exists R.B$.

Finally, given $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$, we define $\texttt{upper}(\mathcal{K}) = \langle \bigcup_{\alpha \in \mathcal{O}} \delta'(\alpha), \mathcal{A} \rangle$.

Theorem 5.1. *Let $\mathcal{K}$ be a satisfiable $\mathcal{ALCHOIQ}^+$ KB and $\mathcal{K}' = \texttt{upper}(\delta(\mathcal{K}))$. Moreover, let $q(\vec{x}) = \exists \vec{y} \varphi(\vec{x}, \vec{y})$ be a CQ. Then,*

(i) *$\mathcal{K}'$ is* RSA$^+$,
(ii) *$cert(q, \mathcal{K}) \subseteq cert(q, \mathcal{K}')$,*
(iii) *if $\vec{x} \in cert(q, \mathcal{K})$ then $\mathcal{P}_{\mathcal{K}',q} \models \texttt{Ans}(\vec{x})$.*

Proof. See the Appendix. □

Example 5.2. Consider, again, our running example (Example 3.1) and $\mathcal{K}'_{ex} = \delta(\mathcal{K}_{ex})$. Let $\mathcal{P}^{ex}_{\text{RSA}}$ be its translation into logic rules (according to Def. 2.3) and M^{ex}_{RSA} its LHM, as in Example 4.2. We know that the dependency graph $G_{\mathcal{K}_{ex}}$ (shown in Fig. 4) is not an oriented forest. Let's assume, again, that Algorithm 1 returns $\{u_4\}$. We `mark` axiom (t8) for constant Skolemization by δ', instead of the standard Skolemization that would be applied by Def. 2.5 (`accepts` is unsafe). We denote the KB resulting from this process with $\mathcal{K}''_{ex} = \texttt{upper}(\mathcal{K}'_{ex})$.

If we consider the query (q2), then:

$$\texttt{cert}\big(q_2, \mathcal{K}''_{ex}\big) = \emptyset. \tag{25}$$

It can be verified that the upper bound computed by PAGOdA is not as tight and results in the following set of answers

$$\big\{\langle journal2, journal3\rangle, \langle journal3, journal2\rangle\big\} \tag{26}$$

[14] This is equivalent to rewriting the axiom as $A \sqsubseteq \exists R.\{c\}$, $\{c\} \sqsubseteq B$.

5.4. Property chain axioms

Our tests show that, general property chain axioms (axioms of type (R4) in Table 1) are quite uncommon in practice. Transitive property axioms, on the other hand, are a specialization of (R4) that can be easily found in common ontologies. While we ignored the presence of these axioms so far, it can be shown that completeness is still guaranteed when including them in the language [14, Theorem 2,Proposition 1]. Intuitively, due to monotonicity of FOL, including more axioms in the computation of the canonical model will lead to a strengthening of the KB. Furthermore, the computational complexity for the computation of the canonical model is still bound by the translation of the problem into Datalog, for which new heuristics have being recently proposed to efficiently handle transitive closure of roles [38]. Note that, in this case, we are not modifying the filtration step, which will then only be able to detect a fraction of the spurious answers, effectively computing an upper bound of the certain answers.

6. Design and architecture

We proposed a new framework to compute CQ answering over unrestricted OWL 2 ontologies by using answer bounds and further refinement steps. The approach has been implemented in a system called ACQuA [42], which, as discussed in the previous sections, offloads different steps in the computation to a selection of underlying systems used as black boxes, i.e., RSAComb, PAGOdA and HermiT.

ACQuA is inspired by the "pay-as-you-go" philosophy that drove the development of PAGOdA and as such shares similarities and capabilities with the latter tool. The idea is to take different steps depending on how the input KB is classified. The input KB needs to go through a consistency check and normalization procedure first. If the normalized KB is inside one of the two ontology languages for which PAGOdA provides full support (i.e., OWL 2 RL and $\mathcal{ELHO}^r_\bot$), we use the PAGOdA lower bound algorithm to compute the answers to the query. This check is purely syntactic over the normalized ontology and can be performed by leveraging the OWLAPI [35] interface for OWL 2 *profile checking*. If the first check fails (i.e., the ontology is not in any of the aforementioned ontology languages), we check whether the ontology is in RSA using RSAComb. If the input ontology is RSA we use the RSAComb algorithm directly (described in Section 2.3.1) and collect the full set of answers to the query. If none of the tractable services for CQ answering are able to capture the KB, we use them to compute lower and upper bound approximations, taking the union of the lower bounds and the intersection of the upper bounds. If the combined bounds match, we have computed a sound and complete set of answers for the input query. If, however, this is not the case, we use PAGOdA's algorithm to compute a subset of the input KB relevant to answer the query, and fall back to HermiT to filter any spurious answers from the gap between the bounds. A summary of these steps was provided in Section 3, along with a visual representation in Fig. 2.

In this section we will describe some design and implementation details that led to the development of ACQuA. In particular, we will focus our attention on RSAComb, a novel implementation of the RSA combined approach for CQ answering, and how the tool can be used to compute lower and upper bounds to the answers of an input query.

6.1. RSAComb

RSAComb [41] is an optimized implementation of the combined approach for CQ answering in RSA. We streamlined and reorganized the algorithm to make the different steps either ontology or query independent. On top of that we designed and implemented an API to introduce approximation capabilities in the system; RSAComb is able to take an unrestricted ontology as input and potentially apply an approximation algorithm (targeting RSA or RSA$^+$) before computing the answers to a query. The system ships with reference implementations of the algorithms for the computation of answer bounds introduced in Sections 4–5.

The system is written in Scala and uses the OWLAPI [35] to interface with the input ontology and manipulate OWL 2 axioms. RDFox is used as an underlying Datalog reasoner; RSAComb has been designed to maximize the amount of computation to be offloaded to RDFox, by redefining problems in terms of queries over a materialized RDF store.

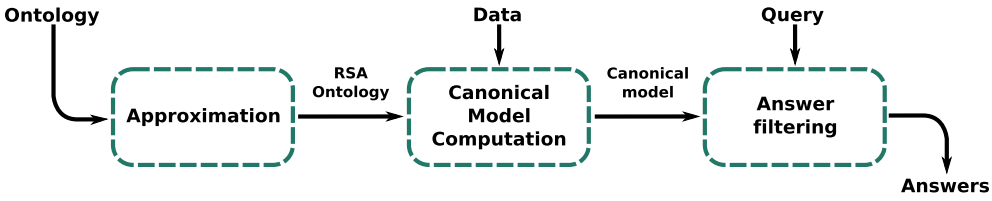


Fig. 5. Workflow of the RSAComb system.

RDFox is used as a black box, and RSAComb can be adapted to use any Datalog reasoner with support for stratified negation and Skolemization. Nonetheless, the use of RDFox allowed us to introduce some optimizations based on particular features provided by the tool.

These are:

- a `SKOLEM` operator,[15] which provides a way to uniquely associate a sequence of terms with a fresh term;
- support for *named graphs* to isolate and cache partial computation;
- support for "TBox reasoning" in order to reason directly on the structure of an ontology even when outside the supported OWL fragment.

We designed and built RSAComb around these general principles:

Modularity The code should be modular and different steps in the algorithm should be as independent of each other as possible. It should be easy to reimplement (or enhance) an intermediate step of the algorithm as long as the *signature* and the *interface* with the system as a whole remain unaltered. We achieved this by an extensive use of Scala *traits*, building a collection of interfaces that describe the behaviour of the different actors that take part in the execution of the combined approach for RSA. As explained in the following sections, the integration with RDFox was also key to providing a good level of modularity to the systems.

Scalability The system has to be able to scale efficiently even for large amounts of data. Partial results are computed when needed and reused whenever possible. A more detailed analysis on the performance and scalability of the system is provided in Section 8.

Integration It should be equally possible to use the system as a self-contained application or integrate it in another system. As such, our software presents a simple but effective command line interface alongside a well-structured set of classes exposing all the necessary tools to work with RSA ontologies, while hiding unnecessary implementation details. The different steps can also be disabled for user convenience.

We will first provide a description of RSAComb as an implementation of the RSA combined approach and then go into details on how lower and upper bound algorithms are implemented in the system.

6.1.1. Overview

Figure 5 summarizes the workflow of RSAComb:

(i) the approximation steps take an unrestricted OWL 2 KB as input and approximate it to a target language handled by the RSA combined approach;
(ii) the canonical model for the resulting RSA KB is computed by materializing the data against a logic program derived from the input ontology;
(iii) a filtering program is derived from the input query and is combined with the canonical model to produce the set of certain answers to the input query over the approximated knowledge base.

The process of importing the input ontology (TBox, RBox) into the system is performed using the OWLAPI. Since importing large amounts of data (ABox) into the system might be expensive, data files are read and data is loaded *on demand* and reused whenever possible to maximize performance.

As mentioned above, two approximation algorithms ship with the system. The first approximation algorithm is an implementation of the algorithm presented in Section 4; it targets the RSA ontology language and maintains soundness w.r.t. CQ answering, i.e., answers to a CQ are a lower bound to the answers to the query over the original

[15] https://docs.oxfordsemantic.tech/tuple-tables.html#rdfox-skolem

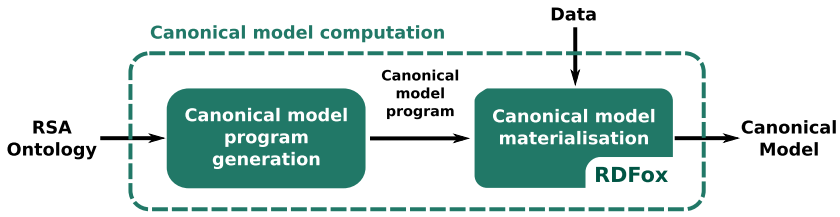


Fig. 6. RSAComb: canonical model computation.

KB. A copy of the ontology, translated into Datalog according to Def. 2.3, is imported into RDFox along with the data and materialized by the reasoner. The dependency graph and equality safety checks (see Definition 2.3) are implemented as queries over the RDF store exposed by RDFox; the original knowledge base is altered accordingly. The second approximation is an implementation of the algorithm introduced in Section 5; it targets RSA$^+$ and maintains completeness w.r.t. CQ answering, i.e., answers to a CQ are an upper bound to the answers to the query over the original knowledge base.

The canonical model is computed for the knowledge base in Step (ii); this is done by converting each axiom in the KB into a logic rule according to Def. 2.5 and uploading it into RDFox. Note that the translation from axioms into logic rules is different from the one in Step (i), hence the need to reload them into RDFox. The data, on the other hand, is safely reused. Finally, the potentially spurious answers to the input query introduced during the canonical model computation are filtered out in Step (iii). It is worth noting that, in this scenario, steps (i),(ii) are *query independent*, while step (iii) is *ontology independent*. As such, when multiple queries are submitted over the same KB, steps (i-ii) are performed "on-demand" and only once, while the third step is performed for each input query.

6.1.2. Canonical model computation

The computation of the canonical model (Figure 6) involves the conversion of the input RSA ontology into logic rules as described in Def. 2.5, and where function symbols are simulated using RDFox's built-in Skolemization feature.

Example 6.1. A Skolemized rule derived from an existential axiom (T5)

$$A(x) \rightarrow R\big(x, f^A_{R,B}(x)\big) \wedge B\big(f^A_{R,B}(x)\big) \tag{27}$$

can be turned into the following RDFox – compatible rule

```
R[?X,?Y], B[?Y] :- A[?X], SKOLEM("A,R,B",?X,?Y).
```

where the built-in operator `SKOLEM` binds `?Y` to a unique value generate from the string "`A,R,B`" and term `?X`.

The system performs the conversion and then offloads the materialization of the rules, combined with the input data, to RDFox.

Since the canonical model is query independent, this process can be performed once and the result cached and reused for every subsequent query over the same input ontology. We achieve this using RDFox's support for RDF named graphs, which enables us to perform operations on specific "named" subsets of the data. Further operations on the graph operate and produce additional data in different named graphs, leaving the materialized canonical model intact.

Axiomatization of $\top$ *and* $\approx$ RDFox has built-in support for $\top$ (`owl:Thing`) and *equality* (`owl:sameAs`), so that $\top$ automatically subsumes any new class introduced within an RDF triple, and equality between terms is always consistent with its semantics.

In both cases we are not able to use these features directly: in the case of top axiomatization, we import axioms as Datalog rules, which are not taken into consideration when RDFox derives new $\top$ subsumptions;[16] in the case of

[16]RDFox accepts both OWL 2 axioms encoded as RDF triples and Datalog rules; these are very different entities in the system and the semantics of special concepts/roles (like $\top$ and $\approx$) is applied to the former.

equality axiomatization, the feature cannot be enabled along other features like *aggregates* and *negation-as-failure* (with the latter extensively used in our system).

To work around this, we introduce the axiomatization for both predicates explicitly. For every concept name $C \in N_C$ and for every role name $R \in N_R$ in the input ontology, we add the following rules to RDFox:

```
owl:Thing[?X] :- C[?X].
owl:Thing[?X], owl:Thing[?Y] :- R[?X,?Y].
```

This gives us the correct semantics for `owl:Thing`.

Similar rules are introduced to axiomatize equality. We make the role *reflexive*, *symmetric* and *transitive*:

```
owl:sameAs[?X,?X] :- owl:Thing[?X].
owl:sameAs[?Y,?X] :- owl:sameAs[?X,?Y].
owl:sameAs[?X,?Z] :- owl:sameAs[?X,?Y], owl:sameAs[?Y,?Z].
```

and introduce substitution rules to complete the axiomatization. For every concept name $C \in N_C$ and for every role name $R \in N_R$ in the input ontology, we add:

```
C[?Y] :- C[?X], owl:sameAs[?X,?Y] .
R[?Z,?Y] :- R[?X,?Y], owl:sameAs[?X,?Z] .
R[?X,?Z] :- R[?X,?Y], owl:sameAs[?Y,?Z] .
```

The `notIn` and `named` predicates Our work also includes a few clarifications on theoretical definitions and their implementation. In the canonical model computation [22], the `notIn` predicate is introduced to simulate the semantics of *set membership* and in particular the meaning of `notIn[a, b]` is "`a` is not in set `b`". During the generation of the canonical model program performed by RSAComb, we have complete knowledge of any set that might be used in a `notIn` atom. For each such set S, and for each element $a \in S$, we introduce the fact `in[`a`,`S`]` in the canonical model. We then replace any occurrence of `notIn[?X,?Y]` in the original program $E_{\mathcal{K}}$ with `NOT in[?X, ?Y]`, where `NOT` is the operator for *negation-as-failure* in RDFox. This is possible because we know that $E_{\mathcal{K}}$ is stratified; moreover the negated predicate introduced in these rules is fully instantiated at program generation and does not appear in the head of any rules, maintaining the stratified structure of the program.

We generate the instances of the predicate `NI`, representing the set of non-anonymous terms in the materialized canonical model, with the following rule:

```
NI[?Y] :- rsa:named[?X], owl:sameAs[?X,?Y] .
```

where `rsa:named` is a predicate representing the set of constants in the original KB.

A final improvement has been made to the computation of the `cycle` function used during the generation of the canonical model program performed by RSAComb. The original definition involved a search over all possible triples (A, R, B) where $A, B \in N_C$ and $R \in N_R$ in the original ontology. We realized that traversing the whole space would significantly slow down the computation, and is *not* necessary; we instead restrict our search over all (A, R, B) triples that appear in a (T5) axiom $A \sqsubseteq \exists R.B$ in the original normalized ontology.

6.1.3. Filtering program and answer computation

As depicted in Fig. 7, answer filtration involves the computation of the filtering program from the input query, the filtering of the materialized canonical model and the final process of gathering the answers.

RSAComb performs the translation of the query into a set of logic rules. This step was modified w.r.t. the original definition [22] to be completely ontology independent by moving the generation of `rsa:named` instances to the canonical model computation step. Furthermore, we redesigned the filtering step to restrict ourselves to use only unary and binary predicates and, as a result, keep the filtering somewhat closer to the realm of description logics (and to the language supported by RDFox). Filtering rules are then greatly simplified by making extensive use of the Skolemization operator provided by RDFox, hence avoiding some expensive *joins* that would result from a standard *reification process*.

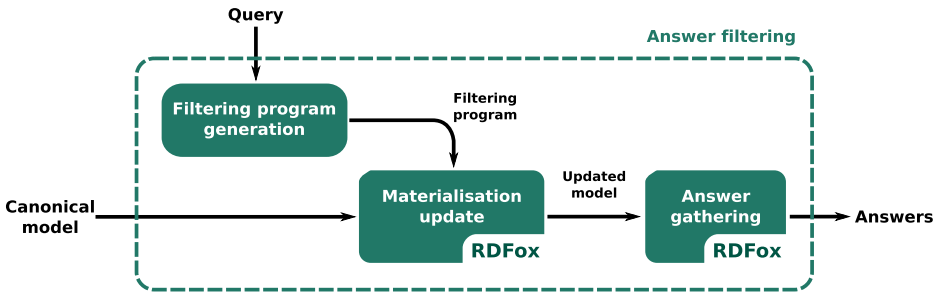


Fig. 7. RSAComb: answer filtering.

Example 6.2. Let $q(\vec{x}) = \psi(\vec{x}, \vec{y})$ be a CQ with $\vec{x} = x_1, \ldots, x_m$, $\vec{y} = y_1, \ldots, y_n$. Rule (3c) of the filtering program (see Table 3) computes the transitive closure of the predicate *id*, keeping track of identity between anonymous terms w.r.t. a specific match for the input query.

$$id(\vec{x}, \vec{y}, u, v), id(\vec{x}, \vec{y}, v, w) \rightarrow id(\vec{x}, \vec{y}, u, w) \tag{28}$$

A standard technique to reduce the arity of predicates is *reification*. Provided we have access to a function KEY to compute a new term that uniquely identifies a tuple of terms, we can *reify* any n-ary atom into a set of n atoms of arity 2. E.g., an atom $P(x, y, z)$ becomes $P_1(k, x)$, $P_2(k, y)$, $P_3(k, z)$, where $k = \texttt{KEY}(x, y, z)$ and P_n, for $1 \leqslant n \leqslant arity(P)$, are fresh predicates of arity 2. Rule (3c) can be *reified* as:

$$\begin{aligned} &id_1(k, x_1), \ldots, id_{m+n}(k, y_n), id_{m+n+1}(k, u), id_{m+n+2}(k, v), \\ &id_1(j, x_1), \ldots, id_{m+n}(j, y_n), id_{m+n+1}(j, v), id_{m+n+2}(j, w), \\ &l := \texttt{KEY}(\vec{x}, \vec{y}, u, w) \rightarrow id_1(l, x_1), \ldots, id_{m+n}(l, y_n), id_{m+n+1}(l, v), id_{m+n+2}(l, w) \end{aligned} \tag{29}$$

The problem with this approach is that it increases the number of joins to be performed to match the body of the rule.

Using the SKOLEM functionality in RDFox, we are able to reduce the arity of a predicate P (*id* in this example) without having to introduce $arity(P)$ fresh predicates (see (30)). The SKOLEM predicate associates a list of terms with a unique blank node; the list of terms and the variable that will be bound to the blank node are passed to the SKOLEM predicate as a single list of arguments. To this end, an atom $id(\vec{x}, \vec{y}, u, v)$ in the original rule becomes an atom $id(k, j)$ of arity 2 where $\texttt{SKOLEM}(\vec{x}, \vec{y}, k)$ and $\texttt{SKOLEM}(\vec{x}, \vec{y}, u, v, j)$ hold, and k and j are bound to two blank nodes uniquely associated with the sequences of terms $\langle \vec{x}, \vec{y} \rangle$ and $\langle \vec{x}, \vec{y}, u, v \rangle$, respectively. Joins over multiple terms (*id* joining over $(\vec{x}, \vec{y})$ in (28)) can now be rewritten into simpler joins (*id* joining over a single term k):[17]

$$id(k, j), \texttt{SKOLEM}(\vec{x}, \vec{y}, u, v, j), id(k, l), \texttt{SKOLEM}(\vec{x}, \vec{y}, v, w, l), \texttt{SKOLEM}(\vec{x}, \vec{y}, u, w, t) \rightarrow id(k, t) \tag{30}$$

The complete rewriting of the filtering program is provided in Table 5. According to the documentation[18] for the SKOLEM operator in RDFox, it can be easily shown that the rewriting is not changing the semantics of the rules, but instead packs and unpacks subsets of variables in order to make rule matching more efficient.

The filtering program is, then, loaded into RDFox and the materialization is updated taking into account the newly introduced rules. The triples produced by this materialization update are stored in a separate named graph to keep the product of filtration separate from the canonical model. This is possible because the signature of the atoms in the head of rules introduced by the filtering program is separate from the signature of the canonical model. When

[17] Rule 30 showcases how the SKOLEM predicate can be used in both directions: given a sequence of terms, we can *pack* them into a single fresh term; given a previously Skolemized term, we can *unpack* it to retrieve the corresponding sequence of terms.

[18] https://web.archive.org/web/20230124163101/https://docs.oxfordsemantic.tech/5.5/tuple-tables.html#rdfox-skolem

Table 5
Improved rules for filtering step for the RSA combined approach

(1)	$\psi(\vec{x}, \vec{y}), \texttt{SKOLEM}(\vec{x}, \vec{y}, s) \rightarrow \texttt{QM}(s)$
(2)	`rsa:named` instances computed in the canonical model step.
(3a)	$\texttt{QM}(s), \texttt{SKOLEM}(\vec{x}, \vec{y}, s), \mathit{not}\ \texttt{NI}(y_i), \texttt{SKOLEM}(i, i, k) \rightarrow \mathit{id}(s, k)$ for each $1 \leqslant i \leqslant \lvert\vec{y}\rvert$
(3b)	$\mathit{id}(s, k_1), \texttt{SKOLEM}(u, v, k_1), \texttt{SKOLEM}(v, u, k_2) \rightarrow \mathit{id}(s, k_2)$
(3c)	$\mathit{id}(s, k_1), \texttt{SKOLEM}(v, u, k_1), \mathit{id}(s, k_2), \texttt{SKOLEM}(u, w, k_2), \texttt{SKOLEM}(v, w, k_3) \rightarrow \mathit{id}(s, k_3)$
(4a)	for all $R(a, y_i), S(b, y_j)$ in q with $y_i, y_j \in \vec{y}$ $R^f(a, y_i), S^f(b, y_j), \texttt{SKOLEM}(i, j, k), \mathit{id}(s, k), \texttt{SKOLEM}(\vec{x}, \vec{y}, s), \mathit{not}\ a \approx b \rightarrow \texttt{fk}(s)$
(4b)	for all $R(a, y_i), S(y_j, b)$ in q with $y_i, y_j \in \vec{y}$ $R^f(a, y_i), S^b(y_j, b), \texttt{SKOLEM}(i, j, k), \mathit{id}(s, k), \texttt{SKOLEM}(\vec{x}, \vec{y}, s), \mathit{not}\ a \approx b \rightarrow \texttt{fk}(s)$
(4c)	for all $R(y_i, a), S(y_j, b)$ in q with $y_i, y_j \in \vec{y}$ $R^b(y_i, a), S^b(y_j, b), \texttt{SKOLEM}(i, j, k), \mathit{id}(s, k), \texttt{SKOLEM}(\vec{x}, \vec{y}, s), \mathit{not}\ a \approx b \rightarrow \texttt{fk}(s)$
	for all $R(y_i, y_j), S(y_m, y_n)$ in q with $y_i, y_j, y_m, y_n \in \vec{y}$
(5a)	$R^f(y_i, y_j), S^f(y_m, y_n), \texttt{SKOLEM}(j, n, k_1), \mathit{id}(s, k_1), \texttt{SKOLEM}(\vec{x}, \vec{y}, s),$ $y_i \approx y_m, \mathit{not}\ \texttt{NI}(y_i), \texttt{SKOLEM}(i, m, k_2) \rightarrow \mathit{id}(s, k_2)$
(5b)	$R^f(y_i, y_j), S^b(y_m, y_n), \texttt{SKOLEM}(j, m, k_1), \mathit{id}(s, k_1), \texttt{SKOLEM}(\vec{x}, \vec{y}, s),$ $y_i \approx y_n, \mathit{not}\ \texttt{NI}(y_i), \texttt{SKOLEM}(i, n, k_2) \rightarrow \mathit{id}(s, k_2)$
(5c)	$R^b(y_i, y_j), S^b(y_m, y_n), \texttt{SKOLEM}(i, m, k_1), \mathit{id}(s, k_1), \texttt{SKOLEM}(\vec{x}, \vec{y}, s),$ $y_j \approx y_n, \mathit{not}\ \texttt{NI}(y_j), \texttt{SKOLEM}(j, n, k_2) \rightarrow \mathit{id}(s, k_2)$
(6)	for each $R(y_i, y_j)$ in q with $y_i, y_j \in \vec{y}$ and $* \in \{f, b\}$ $R^*(y_i, y_j), \mathit{id}(s, k_1), \mathit{id}(s, k_1), \texttt{SKOLEM}(i, v, k_1), \texttt{SKOLEM}(\vec{x}, \vec{y}, s),$ $\mathit{id}(s, k_2), \texttt{SKOLEM}(j, w, k_2), \texttt{SKOLEM}(v, u, k_3) \rightarrow \texttt{AQ}^*(s, k_3)$
	for each $* \in \{f, b\}$
(7a)	$\texttt{AQ}^*(s, k) \rightarrow \texttt{TQ}^*(s, k)$
(7b)	$\texttt{AQ}^*(s, k_1), \texttt{SKOLEM}(u, v, k_1), \texttt{TQ}^*(s, k_2), \texttt{SKOLEM}(v, w, k_2), \texttt{SKOLEM}(u, w, k_3) \rightarrow \texttt{TQ}^*(s, k_3)$
(8a)	$\texttt{QM}(s), \texttt{SKOLEM}(\vec{x}, \vec{y}, s), \mathit{not}\ \texttt{named}(x) \rightarrow \texttt{sp}(s)$ for each $x \in \vec{x}$
(8b)	$\texttt{fk}(s) \rightarrow \texttt{sp}(s)$
(8c)	$\texttt{TQ}^*(s, k), \texttt{SKOLEM}(v, v, k) \rightarrow \texttt{sp}(s)$ for each $* \in \{f, b\}$
(9)	$\texttt{QM}(s), \mathit{not}\ \texttt{sp}(s), \texttt{SKOLEM}(\vec{x}, \vec{y}, s), \texttt{SKOLEM}(\vec{x}, k) \rightarrow \texttt{Ans}(k)$

processing a new query, the only step we need to take is to drop the named graph associated with the filtration from the previous query, leaving unaltered all other triples. Better yet, here we have the possibility to execute queries in parallel, each one associated with a separate filtering program and hence storing their derivations in different named graphs. The materialization update for each of the queries is isolated and does not interfere with the other processes.

At this point, the task of gathering the answers to the query over the input KB is reduced to querying a materialized named graph for the atoms representing the certain answers.

Example 6.3. Given a query $q(\vec{x}) = \exists\varphi(\vec{x}, \vec{y})$, with $\vec{x} = \langle x_1, x_2, x_3\rangle$, we can retrieve the answers to q with the following query

```
1 SELECT ?x1 ?x2 ?x3
2 WHERE {
3     ?K rdf:type rsa:Ans .
4     TT rdfox:SKOLEM { ?x1 ?x2 ?x3 ?K }
5 }
```

where we first collect all instances `?K` of the class `rsa:Ans`, and then we unpack them at line 4 using the custom RDFox syntax for the `SKOLEM` operator, to retrieve the actual answers. When answering BCQs, we only need to check for an `rsa:Ans` witness, i.e., an instance of `rsa:Ans` in the RDF store:

```
[?X,rdfs:subPropertyOf,?Y], [?Y,rdfs:subPropertyOf,?X] :- [?X,owl:equivalentProperty,?Y].

[?Y,owl:inverseOf,?X] :- [?X,owl:inverseOf,?Y] .
[?Yi,rdfs:subPropertyOf,?Xi] :-
    [?X,rdfs:subPropertyOf,?Y], [?Xi,owl:inverseOf,?X], [?Yi,owl:inverseOf,?Y] .

[?X,rdfs:subPropertyOf,?X], [?Y,rdfs:subPropertyOf,?Y] :- [?X,rdfs:subPropertyOf,?Y].
[?X,:subPropertyOfTrans,?Y] :- [?X,rdfs:subPropertyOf,?Y].
[?X,:subPropertyOfTrans,?Z] :- [?X,:subPropertyOfTrans,?Y], [?Y,:subPropertyOfTrans,?Z].
```

Listing 1. Rules for role subsumption reasoning

```
ASK { ?K rdf:type rsa:Ans . }
```

6.2. *Lower bound approximation to RSA*

As described in Section 4, we propose a novel algorithm for approximating an unrestricted input KB to RSA. The procedure is composed of 3 main steps:

1. Approximation to $\mathcal{ALCHOIQ}$ via axiom filtering;
2. Approximation to Horn-$\mathcal{ALCHOIQ}$ via program shifting;
3. Approximation to RSA by reducing the ontology dependency graph to an oriented forest and ensuring equality safety properties.

The first two steps are entirely carried out by RSAComb in a straightforward way. The knowledge base is first filtered by axiom type and then program shifting (Def. 4.1) is applied to all relevant axioms. The last step is designed to partially offload the task to RDFox; this involves:

- building and reasoning over a custom dependency graph derived from the materialization of the input data over a Horn-$\mathcal{ALCHOIQ}$ KB;
- reasoning over the knowledge base itself, and in particular performing some RBox reasoning task.

We first translate the axioms in the knowledge base according to Definition 2.3, and import them, along with the data, into RDFox. The imported data and its materialization contain all instances of the atom `E`, used to build the dependency graph for the input ontology. After retrieving all instances of `E`, querying the RDFox triple store with the following query

```
SELECT ?X ?Y WHERE { ?X rsa:E ?Y }
```

RSAComb builds the dependency graph for the input KB. Using Algorithm 1 we detect and break cycles by iteratively removing nodes. The existential axioms corresponding to the nodes returned by the visit are removed from the input ontology.

For the equality safety check we need to reason over the ontology itself and in particular perform some reasoning over its RBox. Regardless of the support offered by the Datalog reasoner for this task, axioms in a knowledge base can be encoded as RDF triples.[19]

RDFox supports importing OWL 2 axioms and the conversion into RDF triples is performed automatically. RBox reasoning (Listing 1) is then achieved by importing the following rules into the RDF store.

These encode reflexivity and transitivity of sub-role axioms (R2) (lines 7–9), taking into account inverse (lines 3–5) and equivalent roles (line 1), as well.

Once both the data and the axioms have been imported and materialized according to their respective rules, the equality safety condition (i) of Definition 2.3 can be formulated as a query (see Listing 2).

[19] https://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/

```
SELECT ?A ?S ?B WHERE {
    ?W owl:sameAs ?T .
    filter ( ?W != ?T ) .
    ?T ?R [ a rsa:U ] .
    ?R rdfs:subPropertyOf ?Si .
    ?Si owl:inverseOf ?S .
    ?X rdf:type owl:Restriction .
    ?X owl:onProperty ?S .
    ?X maxQualifiedCardinality "1" .
    ?X owl:onClass ?B .
    ?A rdfs:subClassOf ?X .
}
```

Listing 2. Condition 1 of equality safety in RSA definition

```
SELECT ?R ?P WHERE {
  ?A ?R ?U .
  ?U ?S ?A .
  ?A a rsa:NI .
  ?U a rsa:U .
  ?R rdfs:subPropertyOf ?P .
  FILTER ( ?R != ?P ) .
  ?P rdfs:subPropertyOfTrans ?T .
  ?T owl:inverseOf ?Ti .
  ?S rdfs:subPropertyOfTrans ?Ti .
}
```

Listing 3. Condition 2 of equality safety in RSA definition

For each pair of atoms $w \approx t$, with w and t distinct, and $R(t, u^A_{R,B})$ (lines 2–4) in M_{RSA} and each role S s.t. $R \sqsubseteq \mathit{Inv}(S)$ (lines 5–6), we query for the tuple $\langle A, S, B\rangle$ such that $A \sqsubseteq \leqslant 1S.B$ is part of the input KB (lines 7–11). For each triple $\langle A, S, B\rangle$ returned by the query we can remove the corresponding axiom (T4) from the input ontology.

Similarly, condition (ii) can be formulated as a query (see Listing 3).

For each pair of atoms $R(a, u^A_{R,B})$, $S(u^A_{R,B}, a)$ in M_{RSA} with $a \in N_I$ (lines 2–5), we detect roles R, S such that there exists a role T for which $R \sqsubseteq^*_{\mathcal{R}} T$ (lines 6–8) and $S \sqsubseteq^*_{\mathcal{R}} \mathit{Inv}(T)$ (lines 9–10). Note that, when detecting $R \sqsubseteq^*_{\mathcal{R}} T$ we "isolate" the first step of the `subPropertyOf` chain (line 6) and query for that couple of roles $\langle R, P\rangle$. In this case the returned couple $\langle R, P\rangle$, identifies an axiom of type (T2) whose removal will break a chain of sub-properties from R to T, making the knowledge base equality safe.

6.3. Upper bound approximation to RSA

The approximation algorithm proposed in Section 5 is implemented in a similar way. Again, the procedure is divided into the following steps:

1. rewriting of $\bot$ into a new nullary predicate $\bot_f$ with no predefined meaning,
2. rewriting of disjunctive rules to eliminate disjunction, and
3. approximation to RSA$^+$.

As discussed before, the first step is not performed in practice. During the computation of the KB approximation and the upper bound set of answers, we simply ignore the satisfiability of the KB. Note that, even if $\bot$ is derived during the process of materialization, RDFox will not derive the entire Herbrand base, to keep the operation as efficient as possible. We can use this to our advantage and still compute a meaningful upper bound approximation.

Table 6
Decidability of CQE for a selection of ontology languages

Ontology language	Combined complexity	Data complexity
$\mathcal{ELHO}^r_\bot$	NP–complete [72]	P–complete [69,72]
OWL 2 QL	NP–complete [10,12]	AC^0 [10,12]
OWL 2 EL	PSPACE–complete [48,72]	P–complete [48,72]
OWL 2 RL	NP–complete [1,29]	P–complete [1,29]
RSA	NP–complete [22]	P–complete [22]
Horn-$\mathcal{ALCHOIQ}$	EXPTIME–complete [13]	P–complete [13]
$\mathcal{SHOIQ}$	open	open
$\mathcal{SROIQ}$	open	open

The rewriting of disjunctive rules is also straightforward, and is performed directly by RSAComb. The choice function is implemented as in PAGOdA, in order to avoid the derivation of $\bot$ (see Section 5.2).

Finally, the third step involves the same framework introduced in the previous section for the lower bound computation, and in particular the construction of the dependency graph and role subsumption reasoning are performed in the same way. Both the enforcing of equality safety and the reduction of the dependency graph to a forest involve a rewriting of the KB according to Def. 5.2, and are implemented directly in RDFox.

Finally, RSAComb can be used to run the combined approach algorithm on the resulting RSA^+ KB. According to Theorem 5.1 the answers produced by RSAComb are an upper bound to the answers to the query.

7. Related work

Conjunctive query answering over knowledge bases is one of the foundational problems when reasoning over ontologies. This, along with its corresponding decision problem (*conjunctive query entailment*, CQE) have been analysed both from the theoretical point of view (with extensive research on its computational complexity) and from the practical point of view, leading to a number of algorithms and their implementation in various reasoning systems. For a summary of the complexity results on decidability of CQE for some of the ontology languages mentioned in this work, see Table 6.

We will next provide an overview of the techniques proposed in the literature to perform CQ answering and in particular look at those tools that make use of bounds computation in order to drive the query execution. A closer comparison of ACQuA with these tools is also provided.

7.1. Query answering techniques

Support for CQ answering is offered natively by several existing reasoners. Some of them achieve this by ensuring sound and complete answers for a specific semantics over a certain family of ontology languages, while others limit the language in which the queries can be expressed. We will now give an overview of the several CQ answering techniques present in the literature.

7.2. Reduction to entailment checking

The first technique we are going to discuss is based on the reduction of CQ answering to entailment checking. Tableau–based DL reasoners like Pellet [71], HermiT [24], RacerPro [31] construct a finite structure that represents a model for the input KB and use *blocking conditions* to ensure the termination of the procedure. These reasoners usually target standard reasoning tasks and only offer limited support for CQ answering. Still, internalisable CQs can be rolled-up and included in the KB, effectively reducing CQ answering to entailment checking of a fresh concept entailed by the rolled-up query.

Pellet [71] provides support for CQ answering under ground semantics and supports CQ answering under certain answer semantics limited to tree-shaped queries (which can be internalized using the rolling-up technique).

HermiT [24] is a fully-fledged reasoner for OWL 2, based on the *hypertableau calculus* [59]; it does not provide a direct interface to answer CQs but reduction to entailment checking can be manually performed as a preliminary step.

RacerPro is a tableau–based system for the $\mathcal{SHIQ}$ DL language; it implements a technique for instance retrieval, called *filter and refine* [82] and is tailored towards KBs with large ABoxes. The idea behind this technique is to first determine obvious (non-)solutions to a concept description (filter) and subsequently perform an optimized instance check (using ABox locality properties) for the remaining individuals (refine). It supports a superset of CQs under ground semantics.

Another tableau-based reasoner, Konclude [77], has been recently adapted to perform CQ answering over expressive ontologies [75], using an absorption-based technique [74,76]. The assertional part of a KB is divided into small packets used to parallelize the model construction of the tableau algorithm. This parallelizable approach, along with the use of caching to avoid the need of synchronization mechanisms between workers, can be used to derive possible answers to a CQ. Candidate answers are then checked using entailment checking, where bindings for the answer variables are restricted to individuals appearing in the possible answers. According to [75], the approach works best when considering ground queries, while the presence of existential variables can require a substantial amount of additional computation.

Overall, the systems described in this section are not primarily designed for CQ answering under certain answer semantics and instead target other reasoning tasks. The technique of reducing CQ answering to entailment checking is supported for expressive ontology languages but may not scale as well as other approaches. Optimizations have been proposed to further limit performance issues; examples are query execution order, based on the input KB [45] and data summarization [17].

When considering the development of fully-fledged reasoners targetting OWL 2, such as HermiT and Konclude, improvements on these reasoners can translate into improvements for hybrid systems like ACQuA and PAGOdA, which directly use these tools as black boxes.

7.3. Materialisation-based reasoners

Materialization-based reasoners are also widely in use and implement the *forward chaining* algorithm on top of (some fragment of) Datalog. Materialization-based systems are often built on top of RDF management systems; i.e., data management systems based on the Resource Description Framework representing knowledge as statements in the form of triples.

Triple stores like Jena [52], Sesame [8] and Virtuoso [21] offer query answering capabilities over RDBMS and support the RDFS description language. OWLim [7] provides support for OWL 2 RL ontologies. A materialisation-based reasoner extensively used in this work is RDFox [60], an RDF store supporting arbitrary Datalog rules over unary and binary predicates. The nature of the tool allows for important optimizations, e.g., incremental updates, and parallel materialization, at the expense of a limited expressivity in the supported description logic language [55–57,60]. RDFox covers most SPARQL 1.1 over an extension of Datalog. There are several other engines that support CQ answering over (extensions of) Datalog; among them, it is worth mentioning DLV [49], which provides support for CQ answering over an extension of disjunctive Datalog.

Although OWL 2 RL is expressive enough to cover a large portion of practical use cases, it lacks some common patterns like *disjunctive knowledge* or *existentially quantified knowledge*, that would potentially render the materialization process either non-deterministic or infinite. Typically, materialisation-based reasoners can still process ontologies outside OWL 2 RL, ignoring axioms that do not fall into the language. Answers to queries are still sound, but might not be complete, effectively providing a *lower bound* to the set of certain answers. This technique is used in the system PAGOdA [85] to effectively compute a sound lower bound to the set of certain answers to a CQ.

7.4. Ontology-mediated query rewriting

DLs are often used to model the domain of interest as collections of concepts and roles. In this sense, ontologies offer a great tool to build high-level semantics on top of some structured data (e.g., relational database).

OBDA directly applies this principle, creating a layer of abstraction on top of an existing data store; an ontology becomes an entry point for the user to access the underlying data via query answering. Another advantage of this approach is that it can rely on the underling data store (e.g., a RDBMS) to carry out the reasoning tasks. The OBDA framework [83] uses an ontology to rewrite an input query (i.e., expanding it by incorporating parts of the ontology). It then uses a set of *mappings*[20] to transform the rewritten query into a query over the underlying relational data sources. The process is called *perfect reformulation* [66] and ensures that the answers to the query over the dataset and the ontology are the same as the answers to the rewritten query over the dataset alone.

It is worth noting that, since the query addresses the data source(s) indirectly, any updates made to the source are immediately reflected into the system. This is in contrast with the materialisation-based approach, where updates in the source require the recomputation (or the update) of the materialized dataset.

The OBDA approach is based on ontologies that fall into the DL-Lite family of DL languages, and hence the OWL 2 QL profile, for which the rewriting of CQs into unions of first-order queries is guaranteed to exist [10]. Perfect reformulation is implemented in QuOnto [2] and further integrated into the MASTRO system [11]. Unfortunately, the query rewriting process can lead to an exponentially larger first-order query [10] and polynomial rewriting is guaranteed only for small fragments of OWL 2, such as OWL 2 QL. For a more in-depth analysis on the performance of the OBDA approach we refer the reader to the Optique project and their work with Equinor [37,44].

The query rewriting technique has been applied to $\mathcal{EL}$ and $\mathcal{ELH}$ [69], showing that UCQs can be rewritten into a Datalog query. The same result does not hold for $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. REQUIEM [64,65] implements a resolution-based query rewriting technique for $\mathcal{ELHIO}^\neg$, a DL covering both DL-Lite and $\mathcal{EL}$. The rewriting is based on the resolution calculus to saturate the set of rules in the ontology and subsequently filter out those containing functional terms. However, the introduction of inverse roles leads to a significant jump in complexity: CQE for $\mathcal{EL}$ and $\mathcal{ELH}$ is NP–complete, whereas it becomes ExpTime–complete for $\mathcal{ELHIO}^\neg$. Depending on the language of the input ontology the rewriting can be a UCQ or a (linear) Datalog query.

We briefly mention the work done in Clipper [19,20] which implements a query rewriting technique for Horn-$\mathcal{SHIQ}$. The rewriting differs from the ones mentioned above since Clipper modifies the dataset as well. A set of inference rules are used to saturate the input ontology and the data is materialized against the Datalog rules in the saturation. The query is rewritten against the subset of existentially quantified rules in the ontology and evaluated against the augmented dataset. The saturated ontology and the rewriting might be exponential in size w.r.t. the input ontology and query. Query rewriting has also been applied to linear Datalog$^\pm$ ontologies (see [61]).

A different approach involves the manipulation and rewriting of the input query [25,28]. The authors propose a decision procedure for CQE for $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ based on the rewriting of the query into a forest shape. By applying the rolling-up technique [36], the problem is reduced to testing the consistency of an extended KB.

7.4.1. Combined approaches

The *combined approach* is another widely known technique for computing a sound and complete set of answers to a CQ. In this scenario the dataset is first augmented by materializing entailed facts w.r.t. the ontology in order to build a model for the input knowledge base. This process is usually query-independent and performed in polynomial time. Spurious answers are then systematically identified by means of a filtration step or by rewriting the query [47,51] in order to derive the certain answers to the input query. An example is the combined approach for CQ answering over RSA ontologies (see Section 2.3.1), widely exploited in this work. The technique has been applied to several description logics in the $\mathcal{EL}$ family, such as the extension of $\mathcal{ELH}$ with $\bot$ and range axioms [51] and $\mathcal{ELHO}^r_\bot$ [73], as well as in the DL-Lite family, e.g., DL-Lite$_{\text{horn}}$ with number restriction [47] and DL-Lite$_\mathcal{R}$ [50]. More recently the combined approach has been applied to Horn-$\mathcal{ALCHOIQ}$ [13], the ontology language underlying RSA.

In general these techniques are designed for a specific ontology language and do not support unrestricted OWL 2 ontology. On the other hand, as shown in this work and in PAGOdA, the combined approach can be easily used as an intermediate step in the computation.

[20] Often expressed in the W3C standard R2RML language [15].

7.4.2. Hybrid approaches

We will now look at tools that combine more than one technique described above to implement CQ answering.

Hydrowl [78] is a reasoner for CQ answering combining an OWL 2 RL reasoner, a query rewriting system and a fully-fledged OWL 2 reasoner. Hydrowl uses a *repairing* strategy [79] (limited to those ontologies for which a repairing exists) and query rewriting to answer an input query q. First a *query base*, i.e., a set of atomic queries that can be answered using the OWL 2 RL reasoner, is derived from the query. It is checked whether the query base "covers" the query, and in that case the OWL 2 RL reasoner is used to answer the query; otherwise the tool falls back to the fully-fledged reasoner. Further investigation on the computation of the query base [85] shows that the algorithm is not always able to automatically extract a set of atomic queries, thus compromising the correctness of the approach.

Absorption-based query entailment checking [74] (inspired by the absorption technique introduced by Steigmiller et al. [76]) also falls into the category of hybrid approaches. An input query is rewritten in order to make its entailment more efficiently detected by the model constructed using an extended version of the tableau algorithm. In this sense, the rewritten query is used to identify the individuals that are involved in the entailment of the query and, at the same time, to guide the construction of the model in the tableau algorithm. The technique is sound for CQE for expressive ontology languages, such as $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$.

PAGOdA [85] uses a hybrid technique to compute the answers to a query, mixing different approaches. The idea is to compute lower/upper bound approximations to the answers to a query by approximating the input ontology into a less expressive language and possibly provide a fallback (more expensive) algorithm to process the answers in the gap between the bounds. For a more detailed description of the approach see Section 2.2 and [85]. ACQuA builds on top of these techniques.

7.4.3. Ontology approximation

The idea of approximating an expressive language into less expressive (but more tractable) languages has been exploited before. This was first introduced by Selman and Kautz [70] and Del Val [81] in the context of logic theories (both propositional and FOL) and has been applied in the context of ontologies and CQ answering as well. Besides PAGOdA, some of the systems that use ontology approximation to explore and restrict the set of answers to a given CQ are SCREECH [34], TrOWL [80] and SHER [17].

The SCREECH system [34] is able to compute an (unsound or incomplete) approximation of the answers to a query under ground semantics. It achieves that by performing a query dependent (and possibly exponential) rewriting of the input $\mathcal{SHIQ}$ ontology to disjunctive Datalog first, and then further to Datalog. Compared to ACQuA, SCREECH can only handle CQ answering under ground semantics over $\mathcal{SHIQ}$ ontologies.

TrOWL [80] is a system providing CQ answering capabilities over OWL 2 DL. It uses a semantic approximation [63] technique to transform an OWL 2 DL ontology into OWL 2 QL for CQ answering and a syntactic approximation [67] from OWL 2 DL to OWL 2 EL for TBox reasoning. While being sound and complete for CQ answering, approximations steps in TrOWL are ontology *and* query dependent, making in harder to reuse partial results in the computation. Moreover, the semantic approximation requires the use of a fully-fledged reasoner to compute a KB approximation whose axioms are valid w.r.t. the input ontology.

The SHER [17] system is a tableau-based reasoner for $\mathcal{SHIN}$ which provides instance retrieval capabilities. The system uses a summarization technique to compute an upper bound to the answers to an instance query. Spurious answers are then filtered out by a following relaxation step [16,17]. Again, this system is sound and complete for instance CQ answering for ontologies within the $\mathcal{SHIN}$ DL language.

In addition, a way to approximate an OWL 2 ontology into an OWL 2 QL ontology maintaining completeness for instance queries is proposed as part of the filter and refine technique presented by Wandelt et al. [82]. The idea is to transform every axiom $C \sqsubseteq D$ in an OWL 2 ontology into a stronger OWL 2 QL axiom $C' \sqsubseteq D'$ such that C subsumes C' and D' subsumes D. The technique is, however, non-deterministic in nature and the approximation can sometimes lead to an unsatisfiable ontology.

Under the umbrella of approximate reasoning for CQ answering, the *query extension* technique [26,27] is of particular relevance. This algorithm aims at improving the bounds of the answers by extending the query with additional atoms obtained analysing the input ontology. The resulting query can then be used to restrict the bounds of subqueries of the initial query.

Table 7
Benchmarks statistics, with LUBM/UOBM data generators depending on a parameter n

	# Axioms	# Facts	# Queries
LUBM(n)	93	$n \times 10^5$	35
UOBM(n)	186	$2.6n \times 10^5$	20
Reactome	559	1.2×10^7	130
Uniprot	442	1.2×10^8	240

Finally, different notions of approximation for ontology-mediated queries over a selection of expressive languages like $\mathcal{ALC}$ and $\mathcal{ALCI}$ have been explored [32]. The authors aim at designing polynomial time approximations towards tractable languages like $\mathcal{ELI}$ or some restricted classes of TGDs "from below and from above" (lower and upper bounds) with respect to CQs (and other query formalisms as well).

8. Evaluation

We provide here an extensive evaluation over a range of benchmark ontologies. We start by looking at some performance results for RSAComb [41], our implementation of the combined approach for RSA, followed by a comparison of our system ACQuA [42] with PAGOdA.[21] Section 8.1 provides an in-depth description of the benchmarks used for the evaluation. Ontologies, data, queries, and scripts used to run tests and generate the graphs shown in this section are freely available online [43].

All experiments were performed on an Intel(R) Xeon(R) CPU E5-2640 v3 (2.60GHz) with 16 real cores, extended via hyper-threading to 32 virtual cores, 512 GB of RAM and running Fedora 33, kernel version 5.10.8-200.fc33.x86_64. We were able to make use of the multicore CPU and distribute the computation across cores, especially for intensive tasks offloaded to RDFox.

8.1. Benchmarks and tools

We use two different sets of benchmark ontologies:

- the *PAGOdA batch* mimics the evaluation process originally performed for PAGOdA [85];
- the *Oxford Ontology Repository (OOR) batch* is a subset of the Oxford Ontology Repository,[22] and it's used to provide a broader evaluation on a wide range of ontology benchmarks.

The PAGOdA batch consists of a selection of ontologies and benchmark data that comes with the PAGOdA distribution.[23] These resources include ontology, data, and queries for:

- LUBM and UOBM, standard benchmarks with a data generator (depending on a numerical parameter) and sample queries. When referring to a dataset generated for a particular parameter we will use LUBM(n) and UOBM(n) for some number n. PAGOdA provides an additional set of queries more challenging for the tool.
- Reactome and Uniprot, realistic ontologies for which both data and relevant queries are provided. To test scalability, the datasets of these ontologies have been sampled in subsets of increasing size.

A summary of the statistics regarding each of these ontologies can be found in Table 7 where n is the parameter passed to the data generator for LUBM and UOBM.

This batch aims at testing the system with a set of well–established benchmark queries. These queries have been defined to stress a system on a broad range of aspects of an answering routine, and vary a lot in terms of complexity and number of answers.

[21] https://github.com/KRR-Oxford/PAGOdA (commit `8651164c`).
[22] http://krr-nas.cs.ox.ac.uk/ontologies/UID/
[23] https://www.cs.ox.ac.uk/isg/tools/PAGOdA/2015/jair/

For the OOR batch we selected 126 ontology from the repository with non-empty ABoxes. A summary of the statistics of the ontologies in the repository can be found online.[24]

Since the Oxford Ontology Repository does not provide any test queries, we generated, for each ontology, a set of sample queries by extracting atomic concept, atomic role and existential patterns from the structure of the ontology. In order to generate a suitable number of queries we used the following step:

1. Import the ontology into RDFox as RDF triples.
2. Query for a specific pattern in the ontology, e.g.,

```
SELECT DISTINCT ?Y ?Z
WHERE {
  ?X rdf:type owl:Restriction ;
     owl:onProperty ?Y ;
     owl:someValueFrom ?Z .
}
```

   to retrieve all existential axioms in the ontology.
3. Convert those patterns into queries.

Using this method, we extracted 14135 concept atomic queries, 4434 role atomic queries and 3893 existential queries for a total of 22462 queries over 126 ontologies. Apart from the basic atomic patterns, we included existential queries of the form

```
SELECT ?X WHERE { ?X <property> [ rdf:type <class> ] }
```

By doing so, we aimed for an empirical confirmation of our ability to produce the correct set of answers under certain answer semantics, for a set of queries that potentially provide different results when considering them under ground semantics.

While the generated queries have a limited number of atoms, the primary aim of this batch is to stress the system with a high number of queries per ontology; this allows us to draw conclusions on the behaviour of the system on a wider variety of test cases.

The collection of SPARQL queries and the scripts to generate them are part of our benchmark distribution [43].

8.2. PAGOdA batch

We now present the test result obtained using the first set of benchmarks. We first tested RSAComb as a standalone system, in order to evaluate its performance and scalability. Later we compare the performance of ACQuA against the original PAGOdA. This is particular usefully since we were able to draw a very close comparison between the two tools and improve upon the observations provided by PAGOdA [85]. This also helped us identify how and when ACQuA's algorithm outperformed PAGOdA, solving some performance issues with the latter tool.

8.2.1. RSAComb

As part of this work, we introduced RSAComb, an improved implementation of the combined approach algorithm for RSA, released as free and open source software [39]. Given that the original reference implementation [22] was not available when we started this work, and some details about the testing process are not provided, we will not try to draw a comparison between the results provided here and those provided in the original paper.

Our implementation is written in Scala and uses RDFox[25] as the underlying Datalog reasoner. At the time of writing, development and testing have been carried out using Scala v2.13.5 and RDFox v5.2. We can easily interface Scala with Java libraries and in particular the OWLAPI [35] for easy ontology manipulation. Thanks to the Java wrapper API provided with RDFox we were able to take advantage of a tight integration with the tool and simplify the following integration into ACQuA.

[24] http://krr-nas.cs.ox.ac.uk/ontologies/readme.htm
[25] https://www.oxfordsemantic.tech/product

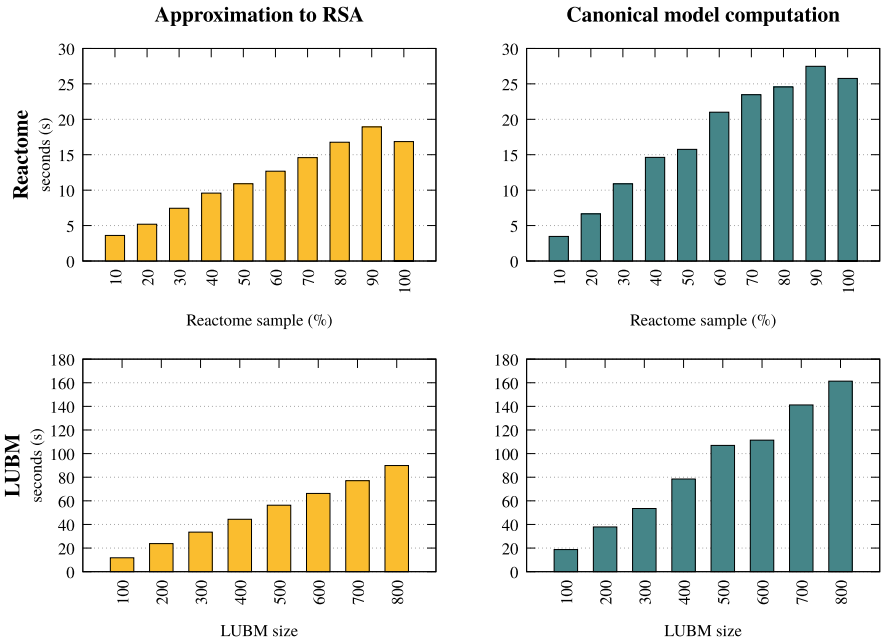


Fig. 8. Scalability of approximation to RSA and canonical model computation in RSAComb.

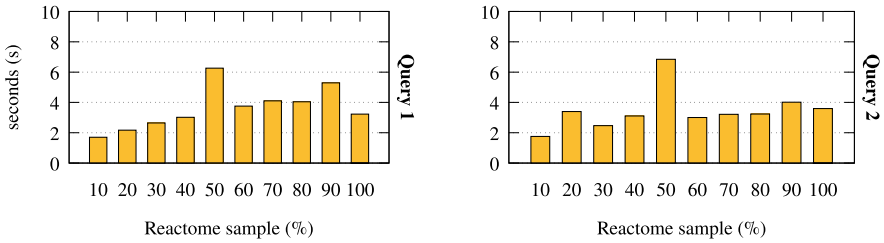


Fig. 9. RSAComb answer filtering in Reactome.

In the following we provide tests result of our system against LUBM [30] and Reactome[26] using the set of queries originally used in the testing of the combined approach for RSA [22]. All results provided below are averages of at least 3 measurements.

In Fig. 8 we show the scalability of our algorithm for the lower bound approximation to RSA and the computation of the canonical model for the approximated ontology. The two steps are query independent and the trend appears to be linear w.r.t. the dataset size, both in LUBM and Reactome; this can be explained by observing that the approximation algorithm involves the materialization of the input dataset against a modified version of the ontology, hence depending on the size of the whole KB;

The filtering process is instead less dependent on the size of the data and more dependent on its composition and distribution. As such, a bigger dataset does not necessarily correspond to a greater amount of filtering, as shown in Fig. 9, where we reported the execution time for query 1 and 2 in Reactome. This figure also shows how the filtering depends on the data distribution; both queries take longer on a 50% sample of the data than on other datasets (even larger ones) due to its specific content. In general, we noticed that the time spent by the system on the filtering step is considerably lower than the time spent on the canonical model computation (as described below, and shown in Fig. 11).

[26] https://elixir-europe.org/platforms/data/core-data-resources

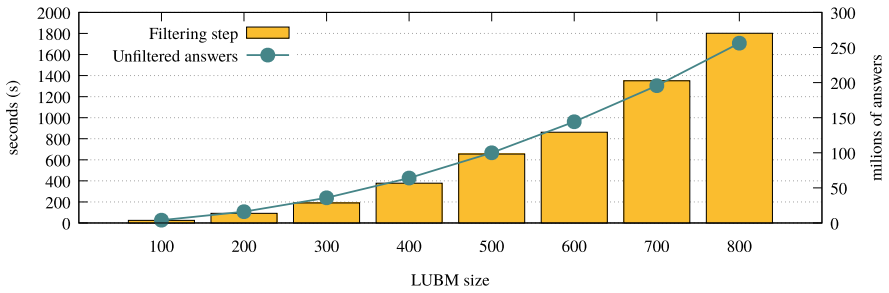


Fig. 10. Answer filtering in Query 2 in LUBM.

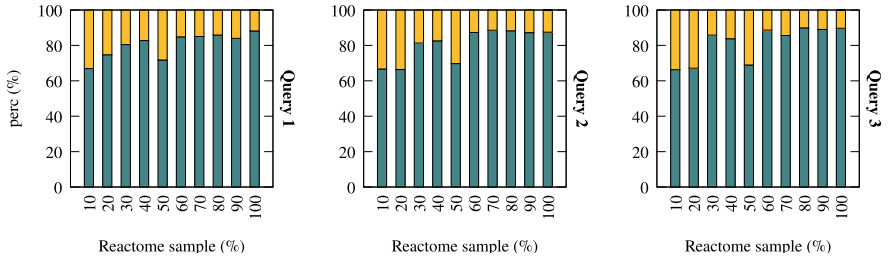


Fig. 11. Percent time distribution of canonical model computation (at the bottom, in blue) and answer filtering (at the top, in yellow) in Reactome.

This unpredictability of the filtering step can "backfire" when a huge amount of filtering is involved. In Fig. 10 we show the filtering time for query 2 in LUBM along with the amount of unfiltered answers that the filtering program needs to process. It is worth noting that less than 1‰ of the unfiltered answers are found to be part of the certain answers. Figure 10 confirms the previous claims that the filtering step grows proportionally to the amount of filtering that is needed for a particular query. Finally, this figure shows how our system is able to handle a gigantic filtering step, processing hundreds of millions of facts in a reasonable amount of time.

Finally, Fig. 11 shows how execution time is distributed among the two main tasks of the combined approach. Filtering takes consistently less that 20% of the total execution time, when considering bigger datasets. As mentioned before, we can limit the impact of the canonical model computation by computing it "offline" whenever we find ourselves in a scenario in which we need to perform query answering over a *fixed* ontology.

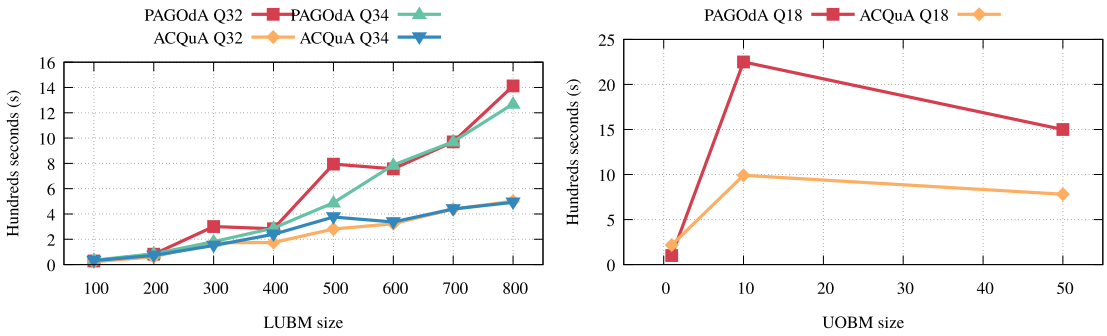
8.2.2. ACQuA

We will now provide test results for the PAGOdA batch against ACQuA; this will allow us to draw a direct comparison between the tool and PAGOdA [85], for which the same benchmarks were used during the evaluation phase. During our tests we were able to reproduce the results provided in the original paper except for UOBM, for which PAGOdA does not terminate with a timeout of 10h and outputs no relevant information.

We chose this as a first set of benchmarks because we were able to use the extensive analysis on PAGOdA's performance to guide our research and easily detect those cases that our system could improve. PAGOdA initially divided its test results into three groups:

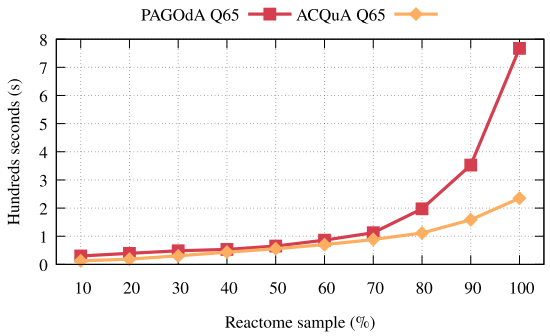
(G1) queries for which the bounds match;
(G2) queries with a non-empty gap, but for which summarization is able to filter out all remaining spurious answers;
(G3) queries where HermiT is called on at least one of the test datasets.

When considering RSAComb and PAGOdA, the building blocks of ACQuA, separately, efficiency in the two tools mainly depends on the input ontology and the type of query answered, with PAGOdA showing worse performance when heavily relying on HermiT. On the other hand, when combining the tools into ACQuA, RSAComb is able to further limit the occurrence of these cases, providing better performance overall.



(a) LUBM query processing.

(b) UOBM query processing.



(c) Reactome query processing.

Fig. 12. Scalability of query processing times for LUBM, UOBM and Reactome in ACQuA vs PAGOdA.

In general ACQuA is able to match PAGOdA's results in all queries in the (G1-2) groups. This should not come as a surprise, since the already excellent results from PAGOdA weren't leaving much room for improvement and were showing that more complex CQ answering techniques were not needed for these families of queries. In particular, for query in the G1 group, ACQuA does not perform any additional step other than PAGOdA's computation of lower and upper bounds (avoiding the use of HermiT altogether).

For this reason we will be focusing on those queries falling in the (G3) group, for which PAGOdA's performance does not scale well.

According to [85, Section 10.3.2], and partially confirmed by our tests, PAGOdA falls back to HermiT in the following queries to compute the correct set of answers: queries 32 and 34 in LUBM, query 18 in UOBM (for some data sizes) and query 65 in Reactome. Figure 12 sums up the results for our tests. Pre-processing times for the ontology are not taken into account here since the process is common to both tools and, in general, can be computed offline.

LUBM For both queries we were able to compute matching bounds, skipping the call(s) to HermiT altogether. This resulted in a significant improvement on the query processing time (Fig. 12a). It is interesting to notice that in this case the nature of the data and the queries seem to lead to a linear growth with respect to the size of the data.

Additionally, while testing the tools, we noticed that PAGOdA was having some difficulties returning a sound set of certain answers to queries involving existential knowledge, potentially falling back to ground semantics instead.

Example 8.1. LUBM TBox contains the following axiom describing the fact that each research assistant works for at least one research group

$$\texttt{ResearchAssistant} \sqsubseteq \exists \texttt{worksFor.ResearchGroup} \tag{31}$$

The following query

Table 8
PAGOdA and ACQuA statistics on OOR batch (over 126 ontologies and 22462 queries)

	Ontologies processed	Queries executed	Non-empty queries
PAGOdA	103	18235	1455
ACQuA	126	22462	2256

```
SELECT ?X WHERE {
    ?X a lubm:ResearchAssistant .
    ?X lubm:worksFor [ rdf:type lubm:ReseachGroup ]
}
```

should return all 39 instances of `ResearchAssistant` contained in LUBM(1), but PAGOdA returns 0 answers (which is only correct under ground semantics).

UOBM We could not perform a direct comparison with UOBM since we were unable to reproduce PAGOdA's results [85], with the tool timing out on a 10h run with no relevant output, even for the smaller datasets. Regardless, we were able to observe a recognizable pattern in the results for query 18. In Fig. 12b, we report our results against an estimate of PAGOdA's performance; the estimation was carried out by looking at the graphs of the original paper and considering the closest values that the resolution of images allowed us to read. Even in this case we were able to avoid the use of HermiT, consequently improving the query processing time overall.

Reactome We were able to answer query 65 with matching bounds, avoiding again the use of HermiT. This resulted in an improvement of almost 600 seconds for the full Reactome dataset.

Furthermore, we found that the answers returned by PAGOdA for some of the queries in LUBM were only correct if considering CQ answering under ground semantics. Examples of these are query 15–16 from the PAGOdA benchmarks, for which PAGOdA was able to return only an incomplete set of answers.[27] In these cases ACQuA was able to fix the issue and compute the sound and complete set of answers under certain answer semantics, by avoiding the use of PAGOdA overall.

8.3. OOR batch

For the second batch of benchmarks executed on the Oxford Ontology Repository, we were able to identify a set of queries for which PAGOdA requires the use of HermiT for the full computation of the query answers.

As shown in Table 8, PAGOdA was able to process 103 out of 126 ontologies considered, executing around 81% of the generated queries; ACQuA, on the other hand, was able to process the entire set of ontologies, answering the full suite of generated queries. Around 10% of the queries have a non-empty answer set, and while ACQuA was able to answer all of them, PAGOdA can reliably answer only $\sim$ 65%.

We identified a set of 18 queries (role atomic queries) over DOLCE [23] for which PAGOdA required the use of HermiT. In these cases, only the lower bound computed by PAGOdA is exact, while ACQuA was able to compute a matching upper bound. This was detected in two different fragments of DOLCE from the repository, corresponding to ontology 14 and 24. Ontology 24 corresponds to the full DOLCE ontology, while ontology 14 is a fragment of 24 partially restricting the ABox. Both ontologies are classified as $\mathcal{SHOIN}(\mathcal{D})$.

In Fig. 13 we provide quantitative and performance results for the queries over ontology 24, where we denote the lower bound, upper bound and query processing time for the corresponding tools with L, U and T respectively. We omit ontology 14 since the results are similar to the ones reported for ontology 24.

In the first 16 queries we obtained comparable results (see Fig. 13). This is understandable since DOLCE is a relatively small ontology (with a very small ABox) and this ended up hiding the performance differences that would potentially appear with larger datasets. Moreover, it should be noted that PAGOdA is able to deal with the larger upper bound by performing a limited amount of calls to HermiT (up to 8). The number of calls increases to 19 in the last two queries; these are also the cases in which we can observe a greater gain in performance using ACQuA.

[27] This is most likely due to a bug in the PAGOdA codebase.

Query	PAGOdA			ACQuA			Total
ID	L	U	T	L	U	T	answers
p127	32	41	17.2s	32	32	16.8s	32
p128	7	14	16.2s	7	7	12.4s	7
p129	7	14	15.9s	7	7	13.9s	7
p154	32	41	16.6s	32	32	17.0s	32
p206	9	19	15.7s	9	9	15.4s	9
p207	9	19	15.8s	9	9	16.5s	9
p210	10	20	15.9s	10	10	14.0s	10
p211	10	20	16.4s	10	10	14.3s	10
p212	12	20	15.8s	12	12	16.5s	12
p213	12	20	15.7s	12	12	16.1s	12
p218	12	20	16.0s	12	12	14.8s	12
p219	12	20	15.8s	12	12	14.7s	12
p266	12	20	16.7s	12	12	12.1s	12
p267	12	20	16.0s	12	12	15.8s	12
p273	12	20	15.9s	12	12	16.1s	12
p274	12	20	16.1s	12	12	15.5s	12
p282	53	77	54.4s	53	53	25.4s	53
p283	53	77	49.7s	53	53	28.0s	53

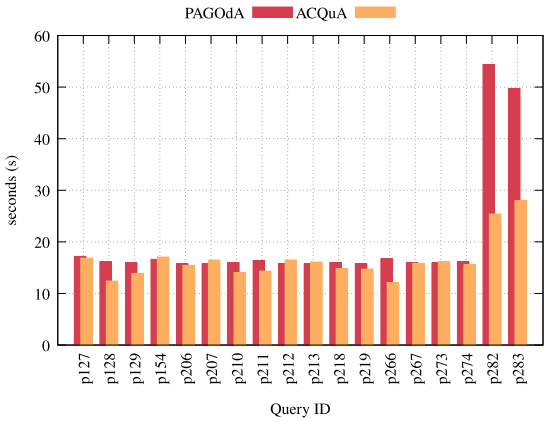


Fig. 13. Execution time (T) on DOLCE queries in PAGOdA (red) vs ACQuA (orange), alongside quantitative results for the lower (L) and upper bounds (U) computed by the two tools.

Finally, we found a set of 23 queries across multiple ontologies for which PAGOdA returned an unsound set of answers. This is the whole set of queries that PAGOdA was unable to process. This behaviour was not observed in ACQuA, which was able to compute the correct answers to the queries, without calling PAGOdA.

For the rest of the tested queries PAGOdA and ACQuA had comparable performance and were able to compute matching bounds.

To conclude this section, we provide a list of performance results and improvements highlighted by our evaluation:

- RSAComb shows linear scalability for preprocessing and canonical model computation steps. Moreover, the filtering time is lower on average than the canonical model computation;
- RSAComb handles huge amounts of data in a reasonable amount of time;
- the additional logic introduced by ACQuA is able to outperform PAGOdA in a variety of test cases, improving *both* the lower and upper bounds;
- ACQuA is able to fix some performance issues present in PAGOdA, by computing matching and hence further limiting the use of HermiT.

9. Discussion and conclusions

In this work, we presented a new hybrid query answering architecture that combines black-box services to provide a CQ answering system for OWL. Our system builds upon scalable CQ answering services for tractable restrictions of OWL, combining them with a CQ answering routine for a more expressive language. The technique is based on the computation of lower and upper bounds to the answers to a CQ and their progressive refinement to compute the full set of certain answers. We proposed two novel algorithms to compute lower and upper bounds to the answers to a query via approximation to RSA and RSA$^+$, respectively, along with reference implementations, used in two new systems:

- RSAComb, an efficient implementation of the combined approach for RSA [22], introducing a new design and the use of RDFox as the underlying Datalog reasoner. The system improves upon the theoretical contribution of the original work by introducing several heuristics, in order to render the algorithm more efficient in practice. The system accepts *any* OWL 2 KB and includes a customizable approximation step to languages compatible with the RSA combined approach. To this end, we included a reference implementation of the novel approximation algorithms for the computation of answer bounds mentioned above.

– ACQuA, a reference implementation of the hybrid architecture combining RSAComb, PAGOdA [85], and HermiT [24] to provide a CQ answering service for OWL. By building ACQuA, we showed how the novel idea of chaining multiple services to refine answer approximations is feasible in practice. Ideally, the services it is built upon can be substituted or paired with more capable ones to improve the system performance–wise.

We provided an extensive evaluation of the systems, first testing scalability and performance of RSAComb as a standalone system and then, comparing ACQuA against PAGOdA. We showed how the additional computational cost introduced by reasoning over a more expressive language like RSA can still provide a significant improvement compared to relying on a fully-fledged reasoner. Additionally, we showed how ACQuA can reliably match PAGOdA's performance, and further limit performance issues originally present in PAGOdA, especially when the tool has to extensively rely on HermiT.

We intend to further extend this work in a few different directions. The RSAComb-based algorithms for the computation of answer bounds depend on a cycle-detection procedure over a KB dependency graph. We think that altering the traversal of the graph and adopting (query dependent) heuristics in the cycle-detection algorithm could improve the quality of the computed bounds.

Moreover, ACQuA mostly focuses on ontology manipulation for computing bounds and further processing gap answers. While query independent processes can be cached or computed offline, a different, complementary, approach would be to study the problem of computing answer bounds from a query perspective. An example of such a technique for computing bounds to answers to SPARQL queries has been presented by Glimm et al. [27].

To conclude, this work led us to believe that relying on hybrid frameworks and leveraging existing systems for CQ answering is a winning strategy that can render the problem more viable in practice. Thanks to its modularity, this approach can benefit from the broader research in the area of knowledge representation, description logics, and CQ answering.

Appendix. Proofs

This chapter provides proofs for lemmas and theorems used in Sections 4–5.[28]

In the following we will consider either an RSA or an RSA$^+$ KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ (and explicitly state when some result holds only for one of the two languages) and a CQ $q(\vec{x}) = \exists \vec{y}.\psi(\vec{x}, \vec{y})$. For $\mathcal{P}_{\mathcal{K},q}$, $E_{\mathcal{K}}$ and $\pi(\mathcal{K})^{\approx,\top}$, we will refer to their LHMs as $\mathcal{M}$, $\mathcal{M}_c$ (*canonical*) and $\mathcal{M}_u$ (*universal*), respectively. Note that, by definition of $\mathcal{P}_{\mathcal{K},q}$, it is the case that $\mathcal{M}_c \subseteq \mathcal{M}$.

We start with the notations concerning terms and atoms. For terms s and t, we write $s \leqslant t$ ($s < t$) iff s is (*strictly*) contained in t. The root of a term t is its non-functional part, i.e., $root(f_1(f_2(\dots(f_n(a))\dots))) = a$. We say that a term t has type (A, R, B) if t is either of the form $v^{A,i}_{R,B}$ or of the form $f^{A}_{R,B}(\cdot)$.

The *derivation level* of a ground atom $a = P(\vec{t}) \in M[\Pi]$ with Π a stratified program, is denoted by $level(a, M[\Pi])$ and is a pair of natural numbers (k, l) where k denotes the stratum of P and l is the smallest number such that $a \in T^{l}_{\Pi_k}(U)$, where $U = \emptyset$, if $k = 1$, and $U = T^{\omega}_{\Pi_{k-1}}(U_i)$, otherwise. The derivation level of a ground term $t \in terms(M[\Pi])$, where Π is a stratified program, is denoted as $level(t, M[\Pi])$ and is a pair of natural numbers (k, l), such that t occurs in an atom $a \in M[\Pi]$ s.t. $level(a, M[\Pi]) = (k, l)$ but t does not occur in any atom $a \in M[\Pi]$ such that $level(a, M[\Pi]) = (k', l')$ and $k' < k$, or $k' = k$ and $l' < l$. When a program Π has only one stratum k, the stratum is dropped from the derivation level of the corresponding atom/term.

Theorem 4.1. *Let $\mathcal{K}' = \langle \mathcal{O}', \mathcal{A} \rangle$ be the $\mathcal{ALCHOIQ}$ restriction of the KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$. Moreover, let $\mathcal{K}'' = \langle \mathtt{shift}(\mathcal{O}'), \mathcal{A} \rangle$. Then $\mathtt{cert}(q, \mathcal{K}'') \subseteq \mathtt{cert}(q, \mathcal{K}')$.*

Proof. Let $\mathcal{M} = M[\pi(\mathcal{K}'')^{\top,\approx}]$. We recall that, given a predicate P in the signature of $\mathcal{K}'$, we denote with $\overline{P}$ a fresh predicate, introduced by $\mathtt{shift}(\cdot)$, intuitively representing the complement of P. In order to prove the theorem, we introduce the following claims:

[28] These proofs are the result of several discussions with the authors of [22].

(i) if $\bot \in \mathcal{M}$, then, $\mathcal{K}'$ is inconsistent;
(ii) if $\overline{P}(c) \in \mathcal{M}$, then, $\mathcal{K}' \not\models P(c)$, for any $\overline{P}$ introduced by `shift` and $\mathcal{K}'$ consistent;
(iii) if $P(c) \in \mathcal{M}$, then, $\mathcal{K}' \models P(c)$, for some P in the signature of $\mathcal{K}'$ and $\mathcal{K}'$ consistent;

We can prove these claims by induction on the derivation level of atoms in $\mathcal{M}$.

(i) If $\bot \in \mathcal{A}$ then, $\mathcal{K}' \models \bot$ and hence $\mathcal{K}'$ is inconsistent. Otherwise, there must be some rule r of the form $\bigwedge_{i=1}^{n} A_i(x) \sqsubseteq \bot$ in $\mathcal{K}''$ such that $A_i(c) \in \mathcal{M}$, for some constant c and $1 \leqslant i \leqslant n$. If $r \in \mathcal{K}''$, then, by definition of `shift`, $r \in \mathcal{K}'$. Moreover, by IH, we have $\mathcal{K}' \models A_i(c)$ for $1 \leqslant i \leqslant n$, and hence $\mathcal{K}' \models \bot$ (i.e., $\mathcal{K}'$ is inconsistent).
(ii) Let $\overline{P}(c) \in \mathcal{M}$, with $\overline{P}$ predicate introduced by `shift` for some predicate P in the signature of $\mathcal{K}'$. Since $\overline{P}(c) \notin \mathcal{A}$, there must be a rule

$$r \equiv \bigwedge_{i=1}^{n} A_i(x) \wedge \bigwedge_{i=1}^{m} \overline{B}_i(x) \rightarrow \overline{P}(x) \tag{32}$$

with $A_i(c) \in \mathcal{M}$ for $1 \leqslant i \leqslant n$ and $\overline{B}_i(c) \in \mathcal{M}$ for $1 \leqslant i \leqslant m$. By IH, $\mathcal{K}' \models A_i(c)$ for $1 \leqslant i \leqslant n$ and $\mathcal{K}' \not\models B_i(c)$ for $1 \leqslant i \leqslant m$. Moreover, if $r \in \mathcal{K}''$ then, there is a rule $r' \in \mathcal{K}'$ s.t. either

(a) r' is of the form $\bigwedge_{i=1}^{n} A_i(x) \wedge P(x) \rightarrow \bigvee_{i=1}^{m} B_i(x)$. Since $\mathcal{K}'$ is consistent, $\mathcal{K}' \not\models P(c)$.
(b) $m = 0$ and r' is of the form $\bigwedge_{i=1}^{n} A_i(x) \wedge P(x) \rightarrow \bot$. Assume $\mathcal{K}' \models P(c)$; then, $\mathcal{K}'$ is inconsistent – contradiction, and hence $\mathcal{K}' \not\models P(c)$

(iii) Let $P(c) \in \mathcal{M}$, for some P in the signature of $\mathcal{K}'$. If $P(c) \in \mathcal{A}$, then $\mathcal{K}' \models P(c)$. Otherwise,

(a) there must be some rule $r \equiv \bigwedge_{i=1}^{n} A_i(x) \wedge \bigwedge_{i=1}^{m} \overline{B}_i(x) \rightarrow P(x)$ in $\mathcal{K}''$, $A_i(c) \in \mathcal{M}$ for $1 \leqslant i \leqslant n$, $\overline{B}_i(c) \in \mathcal{M}$ for $1 \leqslant i \leqslant m$. By IH, $\mathcal{K}' \models A_i(c)$ for $1 \leqslant i \leqslant n$ and $\mathcal{K}' \not\models B_i(c)$ for $1 \leqslant i \leqslant m$. Moreover, if $r \in \mathcal{K}''$, there must be a rule $r' \in \mathcal{K}'$ of the form

$$\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{i=1}^{m} B_i(x) \vee P(x) \tag{33}$$

Since $\mathcal{K}'$ is consistent, $\mathcal{K}' \models P(c)$.
(b) For all other possible rules r that can derive $P(c)$, it is the case that $r \in \mathcal{K}''$ implies $r \in \mathcal{K}'$ and, by IH, we have that $\mathcal{K}' \models P(c)$.

If $q = \bot$, then the theorem follows from claim (i). Otherwise, let $q(\vec{x}) = \exists \vec{y} \varphi(\vec{x}, \vec{y})$ and let σ be a certain answer to q w.r.t. $\mathcal{K}''$. Then, by definition, there exists σ' such that, for every $\alpha \in \varphi(\vec{x}, \vec{y})\sigma\sigma'$, $\alpha \in \mathcal{M}$ and, by claim (iii), $\mathcal{K}' \models \alpha$. Finally, we have that $\mathcal{K}' \models \varphi(\vec{x}, \vec{y})\sigma\sigma'$, and hence $\mathcal{K}' \models \exists \vec{y} \varphi(\vec{x}, \vec{y})\sigma$, which, by definition of conjunctive query answer, implies $\sigma \in \mathtt{cert}(q, \mathcal{K}')$. □

We next relate terms in $\mathcal{M}_c$ and $\mathcal{M}_u$ to terms in M_{RSA}.

Lemma A.1. *Let $\eta_c : terms(\mathcal{M}_c) \rightarrow terms(M_{RSA})$ be the following function*

$$\eta_c(t) = \begin{cases} t & \text{if } t \in N_I \\ u^A_{R,B} & \text{if } t \text{ is of type } (A, R, B) \end{cases} \tag{34}$$

Then, for every $t_1, t_2 \in terms(\mathcal{M}_c)$ it holds that

- $A(t_1) \in \mathcal{M}_c$ *implies* $A(\eta_c(t_1)) \in M_{RSA}$;
- $R(t_1, t_2) \in \mathcal{M}_c$ *implies* $R(\eta_c(t_1), \eta_c(t_2)) \in M_{RSA}$;
- $t_1 \approx t_2 \in \mathcal{M}_c$ *implies* $\eta_c(t_1) \approx \eta_c(t_2) \in M_{RSA}$.

Proof. Trivial, by definition of M_{RSA} and induction on the derivation level of atoms in $\mathcal{M}_c$. □

Lemma A.2. *Let $\eta_u : terms(\mathcal{M}_u) \to terms(M_{RSA})$ be the following function*

$$\eta_u(t) = \begin{cases} t & \text{if } t \in N_I \\ u^A_{R,B} & \text{if } t \text{ is of type } (A, R, B) \end{cases} \tag{35}$$

Then, for every $t_1, t_2 \in terms(\mathcal{M}_u)$ it holds that

- *$A(t_1) \in \mathcal{M}_u$ implies $A(\eta_u(t_1)) \in M_{RSA}$*
- *$R(t_1, t_2) \in \mathcal{M}_u$ implies $R(\eta_u(t_1), \eta_u(t_2)) \in M_{RSA}$*
- *$t_1 \approx t_2 \in \mathcal{M}_u$ implies $\eta_u(t_1) \approx \eta_u(t_2) \in M_{RSA}$*

Proof. Trivial, by definition of M_{RSA} and induction on the derivation level of atoms in $\mathcal{M}_u$. □

Lemma A.3. *Let $t_1, t_2 \in terms(\mathcal{M}_c)$. Then, $\beta \equiv t_1 \approx t_2 \in \mathcal{M}_c$ implies* at least one *of the following holds:*

1. *$t_1 \approx a \in \mathcal{M}_c$, for some $a \in N_I$,*
2. *t_1 is of the form $f(u)$ and t_2 is of the form $g(v)$ with $u \approx v \in \mathcal{M}_c$.*
3. *t_1 is of the form $v^{A,i}_{R,B}$ and t_1 and t_2 are identical (i.e., the same term),*

Proof. We prove the lemma, together with the following additional claims, by induction on the derivation level of atoms in $\mathcal{M}_c$:

(i) Let $R(t_1, t_2) \in \mathcal{M}_c$ with t_2 of some type τ, $R \sqsubseteq^*_{\mathcal{R}} S$ for some S occurring in an axiom (T4). Moreover, let $t_3 \in terms(\mathcal{M}_c)$ s.t. $t_2 \approx t_3 \in \mathcal{M}_c$ with $\eta_c(t_2) \neq \eta_c(t_3)$. Then, t_2 is of the form $f(u)$ with $u \approx t_1 \in \mathcal{M}_c$.

(ii) Let $R(t_1, t_2) \in \mathcal{M}_c$ with t_1 of some type τ, $R \sqsubseteq^*_{\mathcal{R}} Inv(S)$ for some S occurring in an axiom (T4) Moreover, let $t_3 \in terms(\mathcal{M}_c)$ s.t. $t_1 \approx t_3 \in \mathcal{M}_c$ with $\eta_c(t_1) \neq \eta_c(t_3)$. Then, t_1 is of the form $f(u)$ with $u \approx t_2 \in \mathcal{M}_c$.

In the following, let $R(t_1, t_2)$ be an atom in $\mathcal{M}_c$.

(i) Moreover, let t_2 be of some type τ, $R \sqsubseteq^*_{\mathcal{R}} S$ for some S occurring in an axiom (T4) and $t_2 \approx t_3 \in \mathcal{M}_c$ with $t_3 \in terms(\mathcal{M}_c)$ and $\eta_c(t_2) \neq \eta_c(t_3)$.
Then, there must be at least one rule in $E_{\mathcal{K}}$ of the form:

(a) $C(x) \to R(x, f^C_{R,D}(x)) \wedge D(f^C_{R,D}(x))$ with $C(t_1) \in \mathcal{M}_c$ and $t_2 = f^C_{R,D}(t_1).t_1 \approx t_1 \in \mathcal{M}_c$ and hence the claim holds.

(b) $C(x) \to R(x, v^C_{R,D}) \wedge D(v^C_{R,D})$ with $C(t_1) \in \mathcal{M}_c$. This is in contradiction with the fact that R is unsafe, i.e., R occurs in a (T5) axiom and $R \sqsubseteq^*_{\mathcal{R}} S$ with S occurring in a (T4) axiom.

(c) $T(x, y) \to R(x, y)$ with $T(t_1, t_2) \in \mathcal{M}_c$ and $level(T(t_1, t_2), \mathcal{M}_c) < level(R(t_1, t_2), \mathcal{M}_c)$. Since $T \sqsubseteq R \sqsubseteq^*_{\mathcal{R}} S$, with S occurring in an axiom (T4), by IH, the claim holds for $T(t_1, t_2)$. Then it trivially holds for $R(t_1, t_2)$ as well.

(d) $Inv(R)(y, x) \to R(x, y)$ with $Inv(R)(t_2, t_1) \in \mathcal{M}_c$ and $level(Inv(R)(t_2, t_1), \mathcal{M}_c) < level(R(t_1, t_2), \mathcal{M}_c)$. It can be easily shown that $Inv(R)$ fulfils all hypothesis of claim (ii), and, by IH, it follows that t_2 is of the form $f(u)$ with $u \approx t_1 \in \mathcal{M}_c$.

(e) $R(x, y) \wedge y \approx z \to R(x, z)$ and $\exists t$ term s.t. $R(t_1, t), t \approx t_2 \in \mathcal{M}_c$. By IH, t is of the form $f(u)$ with $u \approx t_1 \in \mathcal{M}_c$. Moreover, since t is of the form $f(u)$, by the main claim of Lemma A.3, t_2 must be of the form $g(v)$ with $u \approx v \in \mathcal{M}_c$. Then, the claim holds, since t_2 is of the form $g(v)$ with $v \approx t_1$, for transitivity of $\approx$.

(f) $R(x, y) \wedge x \approx z \to R(z, y)$ and $\exists t$ term s.t. $R(t, t_2), t \approx t_1 \in \mathcal{M}_c$. Similar to case (i)e, using claim (ii).

(ii) Similarly, let t_1 be of some type τ, $R \sqsubseteq^*_{\mathcal{R}} Inv(S)$ for some S occurring in an axiom (T4) and $t_1 \approx t_3 \in \mathcal{M}_c$ with $t_3 \in terms(\mathcal{M}_c)$ and $\eta_c(t_1) \neq \eta_c(t_3)$.
Then, there must be at least one rule in $E_{\mathcal{K}}$ of the form:

(a) $C(x) \rightarrow R(x, f^C_{R,D}(x)) \wedge D(f^C_{R,D}(x))$ with $C(t_1) \in \mathcal{M}_c$ and $t_2 = f^C_{R,D}(t_1)$. Then, from Lemma A.1, it follows that $R(\eta_c(t_1), u^C_{D,R}) \in M_{RSA}$. But then, $\mathcal{K}$ is not equality-safe, since:

- $t_1 \approx t_3 \in \mathcal{M}_c$ with $\eta_c(t_1) \neq \eta_c(t_3)$. Then by definition of η_c in Lemma A.1, t_1, t_2 must be distinct.
- $R(\eta_c(t_1), u^C_{D,R}) \in \mathcal{M}_{RSA}$.
- $\exists S$ s.t. $R \sqsubseteq^*_{\mathcal{R}} Inv(S)$ and S occurs in an axiom (T4).

This contradicts our hypothesis that $\mathcal{K}$ is RSA.

(b) $C(x) \rightarrow R(x, v^C_{R,D}) \wedge D(v^C_{R,D})$ – in contradiction with the fact that R is unsafe.
(c) $T(x, y) \rightarrow R(x, y)$ such that $T(t_1, t_2) \in \mathcal{M}_c$, similar to case (i)c.
(d) $Inv(R)(y, x) \rightarrow R(x, y)$ such that $Inv(R)(t_2, t_1) \in \mathcal{M}_c$, similar to case (i)d and using claim (i).
(e) $R(x, y) \wedge y \approx z \rightarrow R(x, z)$ and a term t_3 such that $R(t_1, t_3), t_3 \approx t_2 \in \mathcal{M}_c$, similar to case (i)e.
(f) $R(x, y) \wedge x \approx z \rightarrow R(z, y)$ and a term t_3 such that $R(t_3, t_2), t_3 \approx t_1 \in \mathcal{M}_c$, similar to case (i)f.

Now, let $\beta \equiv t_1 \approx t_2 \in \mathcal{M}_c$; then, there must be some rule in $E_{\mathcal{K}}$ of the form:

(a) $\top(x) \rightarrow x \approx x$ such that $t_1 = t_2 = x$. We can distinguish three different cases:

- $x = a$, for some $a \in N_I$. Then, $t_1 \approx a$ and condition 1 is satisfied.
- x is of the form $f^A_{R,B}(u)$ for some type (A, R, B). Then, $t_1 = t_2 = f^A_{R,B}(u)$ with $u \approx u$ because of the semantics of $\approx$; condition 2 is satisfied.
- x is of the form $v^{A,i}_{R,B}$ for some type (A, R, B) and $i \in \{0, 1, 2\}$. Then, $t_1 = t_2 = v^{A,i}_{R,B}$ and condition 3 is satisfied.

(b) $A(x) \rightarrow x \approx a$, with $A(t_1) \in \mathcal{M}_c$ Then, $t_2 = a$ and $t_1 \approx a \in \mathcal{M}_c$; condition 1 is fulfilled.
(c) $A(x) \wedge S(x, y) \wedge B(y) \wedge S(x, z) \wedge B(z) \rightarrow y \approx z$ and $\exists t_3$ term, s.t. $A(t_3), S(t_3, t_2), B(t_2), S(t_3, t_1), B(t_1) \in \mathcal{M}_c$. We distinguish between the following cases:

- $\eta_c(t_1) = \eta_c(t_2)$. If $t_1 = t_2$ the claims of the lemma trivially hold. If $t_1 \neq t_2$, then t_1 and t_2 must have the same type (C, R, D). Then $t_1 = f^A_{R,B}(u)$ and $t_2 = f^A_{R,B}(v)$ for some type (A, R, B) and with $u \neq v$. It can be shown that atoms $S(t_3, f^A_{R,B}(u)), S(t_3, f^A_{R,B}(v))$ cannot be introduced in an RSA ontology.
- $\eta_c(t_1) \neq \eta_c(t_2)$. If either $t_1 = a$ or $t_2 = b$, with $a, b \in N_1$, then, condition 1 trivially holds for β. Otherwise, from claim (i) it follows that:

  * $t_1 = f(u)$, with $u \approx t_3 \in \mathcal{M}_c$.
  * $t_2 = g(v)$, with $v \approx t_3 \in \mathcal{M}_c$.

  Then, for transitivity of $\approx$, $u \approx v \in \mathcal{M}_c$ and condition 2 holds for β.

(d) $x \approx y \rightarrow y \approx x$ with $t_2 \approx t_1 \in \mathcal{M}_c$. By IH, the lemma holds for $t_2 \approx t_1$ and, since all conditions are symmetric, it holds for β as well.
(e) $x \approx y \wedge y \approx z \rightarrow x \approx z$ and $\exists t_3$ term s.t. $t_1 \approx t_3, t_3 \approx t_2 \in \mathcal{M}_c$. By IH, the lemma holds for $t_1 \approx t_3$ and $t_3 \approx t_2$:

- If condition 1 holds for $t_1 \approx t_3$, s.t. $t_1 \approx a$ for some $a \in N_I$, then it holds for $t_3 \approx t_2$ (since $t_3 \approx t_1 \approx a$) and for β.
- If condition 2 holds for $t_1 \approx t_3$, then t_1 is of the form $f(u)$ and t_3 is of the form $g(v)$, with $u \approx v \in \mathcal{M}_c$. Since t_3 is of the form $g(v)$, condition 2 must hold for $t_3 \approx t_2$ as well, and hence t_2 is of the form $h(w)$, with $v \approx w \in \mathcal{M}_c$. Then, for transitivity of $\approx$, $u \approx w$ and condition 2 holds for β.
- If condition 3 holds for $t_1 \approx t_3$, then t_1 and t_3 are identical and of the form $v^{A,i}_{R,B}$ for some type (A, R, B) and $i \in \{0, 1, 2\}$. Since t_3 is of the form $v^{A,i}_{R,B}$, condition 3 must hold for $t_3 \approx t_2$ as well, and hence $t_2 = v^{A,i}_{R,B}$. Then, $t_1 = t_2 = v^{A,i}_{R,B}$ and condition 3 holds for β. □

Lemma A.4. *Let $t_1, t_2 \in terms(\mathcal{M}_u)$. Then $t_1 \approx t_2 \in \mathcal{M}_u$ implies that either:*

1. *$t_1 \approx a \in \mathcal{M}_u$, for some $a \in N_I$*

2. *t_1 is of the form $f(u)$ and t_2 is of the form $g(v)$ with f, g function symbols in $\mathcal{M}_u$ and $u \approx v \in \mathcal{M}_u$*

Proof. Similar to the proof for Lemma A.3, by induction on the derivation level of atoms in $\mathcal{M}_u$. □

Definition A.1. Let Ψ be a conjunction of atoms of the form

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} R_j(u_{1j}, u_{2j}) \tag{36}$$

An *adornment* for Ψ is a vector $\vec{a}$ such that $|\vec{a}| = n$ and $a_j \in \{f, b, \lrcorner\}$ for every $1 \leqslant j \leqslant n$ (where $\lrcorner$ denotes the empty adorning, i.e., $R^{\lrcorner}$ is syntactically equivalent to R). We denote with $\Psi^{\vec{a}}$ the adorned formula:

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} R_j^{a_j}(u_{1j}, u_{2j}) \tag{37}$$

where $R_j^{a_j}$ is a syntactic renaming of R_j for every $1 \leqslant j \leqslant n$.

Definition A.2. Let $\Psi^{\vec{a}}$ be the adorned formula of the form

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} R_j^{a_j}(u_{1j}, u_{2j}) \tag{38}$$

Then, the normal form of $\Psi^{\vec{a}}$, denoted with $\Psi_n^{\vec{a}}$, is the formula

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} L_j \tag{39}$$

where

$$L_j = \begin{cases} R(u_{1j}, u_{2j}) & \text{if } a_j = \lrcorner \\ R^f(u_{1j}, u_{2j}) & \text{if } a_j = f \\ Inv(R)^f(u_{2j}, u_{1j}) & \text{if } a_j = b \end{cases} \tag{40}$$

Definition A.3. Let $q(\vec{x}) = \exists \vec{y}\ \psi(\vec{x}, \vec{y})$ be a CQ, $\lambda : terms(q) \to terms(\mathcal{M})$ be a homomorphism and $\vec{a}$ be an adornment for $q = \psi(\vec{x}, \vec{y})$. Then $(\lambda, \vec{a})$ is said to be an *adorned match* for q over $\mathcal{M}$ iff the following conditions both holds:

1. $\mathcal{M} \models (\psi(\lambda(\vec{x}), \lambda(\vec{y})))^{\vec{a}}$;
2. $\forall R(t_1, t_2) \in (\psi(\lambda(\vec{x}), \lambda(\vec{y})))^{\vec{a}}$, we have $R^f(t_1, t_2) \notin \mathcal{M}$ and $R^b(t_1, t_2) \notin \mathcal{M}$.

Definition A.4. Let $(\lambda, \vec{a})$ be an adorned match for $q(\vec{x})$ over $\mathcal{M}$. We say that $(\lambda, \vec{a})$ is *non-anonymous* if $\texttt{named}(\lambda(x)) \in \mathcal{M}$ for all $x \in \vec{x}$.

Definition A.5. Let $(\lambda, \vec{a})$ be an adorned match for $q(\vec{x})$ over $\mathcal{M}$. We say that $(\lambda, \vec{a})$ is *fork-free* iff for every two atoms of the form $R^f(u, y_i), S^f(v, y_j) \in (\psi(\vec{x}, \vec{y}))_n^{\vec{a}}$, such that $y_i, y_j \in \vec{y}$ and $id(\lambda(\vec{x}), \lambda(\vec{y}), i, j) \in \mathcal{M}$, it is the case that $\lambda(u) \approx \lambda(v)$.

Definition A.6. Let $(\lambda, \vec{a})$ be an adorned match for $q(\vec{x})$ over $\mathcal{M}$. We say that $(\lambda, \vec{a})$ is *acyclic* iff there is no sequence of atoms

$$R_{o_1}^f(y_{l_1}, y_{l_2}), R_{o_2}^f(y_{l_3}, y_{l_4}), \ldots, R_{o_p}^f(y_{l_{2p-1}}, y_{l_{2p}}) \in \left(\psi(\vec{x}, \vec{y})\right)_n^{\vec{a}} \tag{41}$$

such that

- $id(\lambda(\vec{x}), \lambda(\vec{y}), l_{2i}, l_{2i+1}) \in \mathcal{M}$ for every $1 \leqslant i \leqslant p$ where $l_{2p+1} = l_1$;
- $\texttt{NI}(\lambda(y_{l_j})) \notin \mathcal{M}$ for every $1 \leqslant j \leqslant 2p$.

Lemma A.5. *For a given substitution $\lambda : \vec{x} \to terms(\mathcal{M})$, it is the case that $\mathcal{M} \models \texttt{Ans}(\lambda(\vec{x}))$ iff there exists an adorned match $(\lambda', \vec{a})$ for q over $\mathcal{M}$ which is non-anonymous, fork-free and acyclic, where λ' is a homomorphism that extends λ to $terms(q)$.*

Proof. Trivial, from the definitions of $\pi(\mathcal{K})^{\approx,\top}$, $\mathcal{M}$ (and in particular the filtering program in Table 5), and Definition A.3. □

Lemma A.6. *For a given substitution $\lambda : \vec{x} \to terms(\mathcal{M})$, if $\lambda(\vec{x}) \in cert(q, \mathcal{K})$ then there exists a match λ' for $q(\vec{x})$ over $\mathcal{M}_u$ where λ' is a homomorphism that extends λ to $terms(q)$.*

Proof. Trivial by the definition of certain answer. □

Definition A.7. Let T'_i be the congruence classes induced by $\approx$ over $terms(\mathcal{M}_u)$, and let t'_i be a collection of terms from $\mathcal{M}_u$ s.t. for every i:

1. $t'_i \in T'_i$;
2. $t'_i \in N_I$ if there exists $t' \in T'_i$ s.t. $t' \in N_I$.

Then, let $\xi : terms(\mathcal{M}_u) \to terms(\mathcal{M}_u)$ be such that $\xi(t) = t'_i$, if $t \in T'_i$ and let $\sigma : terms(\mathcal{M}_u) \to terms(\mathcal{M}_u)$ be a function which has the following properties:

$$\sigma(t) = \begin{cases} \xi(t) & \text{if } \xi(t) \in N_I \\ f(\sigma(u)) & \text{if } \xi(t) = f(u) \text{ for some function symbol } f \text{ in } \mathcal{M}_u \end{cases} \tag{42}$$

Also, let $\theta : terms(\mathcal{M}_u) \to terms(\mathcal{M}_c)$ be the following function:

$$\theta(t) = \begin{cases} t & \text{if } t \in N_I \\ f^A_{R,B}(\theta(u)) & \text{if } t = f^A_{R,B}(u) \text{ and } R \text{ is unsafe} \\ v^{A,0}_{R,B} & \text{if } t = f^A_{R,B}(u),\ R \text{ is safe and } \theta(u) \notin \texttt{unfold}(A, R, B) \\ v^{A,i+1}_{R,B} & \text{if } t = f^A_{R,B}(u),\ R \in \texttt{confl}(R) \text{ and } \theta(u) = v^{A,i}_{R,B}, \text{ for } i = 0, 1 \\ v^{A,1}_{R,B} & \text{if } t = f^A_{R,B}(u) \text{ and } \theta(u) \in \texttt{cycle}(A, R, B) \end{cases} \tag{43}$$

Definition A.8. Given $t \in terms(\mathcal{M}_*)$ with $* \in \{\sqcup, c, u\}$, we define the nesting level of t as

$$depth_*(t) = \begin{cases} 0 & \text{if } \xi(t) \in N_I \\ 1 + depth_*(u) & \text{if } \xi(t) = f(u) \end{cases} \tag{44}$$

with f a function symbol in $\mathcal{M}_*$.

Lemma A.7. *Let σ be as in Definition A.7, and f, h function symbols in $\mathcal{M}_u$. Then, for every $t, t_1, t_2 \in terms(\mathcal{M}_u)$, it holds that:*

1. $\sigma(t) \approx t \in \mathcal{M}_u$
2. $\sigma(f(t)) \approx f(\sigma(t)) \in \mathcal{M}_u$
3. $t_1 \approx t_2 \in \mathcal{M}_u$ *implies* $\sigma(t_1) \approx \sigma(t_2) \in \mathcal{M}_u$
4. $\sigma(f(t)) = h(\sigma(t))$ *or* $\sigma(f(t)) \in N_I$

Proof. Given $t \in terms(\mathcal{M}_u)$:

1. We show by induction over $depth_u(t)$ that $\sigma(t) \approx t \in \mathcal{M}_u$. If $depth_u(t) = 0$, $\sigma(t) = \xi(t)$ and $\xi(t) \approx t \in \mathcal{M}_u$. If $depth_u(t) > 0$, $\sigma(t) = f(\sigma(u))$, where $\xi(t) = f(u)$ and by IH $\sigma(u) \approx u \in \mathcal{M}_u$. Then $f(\sigma(u)) \approx f(u) = \xi(t) \in \mathcal{M}_u$. As $\xi(t) \approx t \in \mathcal{M}_u$, it follows that $\sigma(t) \approx t \in \mathcal{M}_u$.
2. From claim 1, $\sigma(f(t)) \approx f(t) \in \mathcal{M}_u$. Furthermore, as $t \approx \sigma(t) \in \mathcal{M}_u$, it follows that $f(t) \approx f(\sigma(t)) \in \mathcal{M}_u$. Thus, $\sigma(f(t)) \approx f(\sigma(t)) \in \mathcal{M}_u$.
3. Follows from the fact that $\xi(t_1) = \xi(t_2)$, for any $t_1 \approx t_2 \in \mathcal{M}_u$.
4. Assume $\xi(f(t)) = h(u)$. Then, $f(t) \approx h(u) \in \mathcal{M}_u$ and, from Lemma A.4, it follows that $t \approx u \in \mathcal{M}_u$. Then, $\sigma(f(t)) = h(\sigma(u)) = h(\sigma(t))$ or $\sigma(f(t)) \in N_I$. □

Lemma A.8. *Let σ and θ be as in Definition A.7. Then, for every $t, t_1, t_2 \in terms(\mathcal{M}_u)$:*

(1) $A(t) \in \mathcal{M}_u$ *implies* $A(\theta(\sigma(t))) \in \mathcal{M}_c$
(2) $R(t_1, t_2) \in \mathcal{M}_u$ *implies* $R(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$
(3) $t_1 \approx t_2 \in \mathcal{M}_u$ *implies* $\theta(t_1) \approx \theta(t_2) \in \mathcal{M}_c$

Proof. From Lemma A.7 it follows that:

- $A(t) \in \mathcal{M}_u$ implies $A(\sigma(t)) \in \mathcal{M}_u$
- $R(t_1, t_2) \in \mathcal{M}_u$ implies $R(\sigma(t_1), \sigma(t_2)) \in \mathcal{M}_u$

In the following we show by induction on the derivation level of atoms in $\mathcal{M}_u$ that:

(i) $A(t) \in \mathcal{M}_u$ implies $A(\theta(t)) \in \mathcal{M}_c$
(ii) $R(t_1, t_2) \in \mathcal{M}_u$ implies $R(\theta(t_1), \theta(t_2)) \in \mathcal{M}_c$
(iii) $t_1 \approx t_2 \in \mathcal{M}_u$ implies $\theta(t_1) \approx \theta(t_2) \in \mathcal{M}_c$

Let a be an atom in $\mathcal{M}_u$. If $a \in \mathcal{A}$, i.e., a is a fact in $\mathcal{K}$, all three conditions are trivially satisfied since $\theta(t) = t$ for $t \in N_I$. Otherwise,

(i) Let $a = A(t)$. Then, there must be a rule in $\pi(\mathcal{K})^{\approx,\top}$:

  (a) $B(x) \rightarrow R(x, f^B_{R,A}(x)) \wedge A(f^B_{R,A}(x))$ and a term u such that $B(u) \in \mathcal{M}_u$ and $t = f^B_{R,A}(u)$. Then, by IH, $B(\theta(u)) \in \mathcal{M}_c$ and $E_\mathcal{K}$ must contain a rule:

  - $B(x) \rightarrow R(x, f^B_{R,A}(x)) \wedge A(f^B_{R,A}(x))$ if R is unsafe.
    Then, $A(f^B_{R,A}(\theta(u))) = A(\theta(f^B_{R,A}(u))) = A(\theta(t)) \in \mathcal{M}_c$.
  - $B(x) \rightarrow R(x, v^{B,0}_{R,A}) \wedge A(v^{B,0}_{R,A})$ if $\theta(u) \notin \texttt{unfold}(B, R, A)$.
    Then, $A(v^{B,0}_{R,A}) = A(\theta(f^B_{R,A}(u))) = A(\theta(t))$ and $A(\theta(t)) \in \mathcal{M}_c$.
  - $B(x) \rightarrow R(x, v^{B,1}_{R,A}) \wedge A(v^{B,1}_{R,A})$ if $\theta(u) \in \texttt{unfold}(B, R, A)$. Similar to the previous case.
  - $B(v^{B,i}_{R,A}) \rightarrow R(v^{B,i}_{R,A}, v^{B,i+1}_{R,A}) \wedge A(v^{B,i+1}_{R,A})$ if $\theta(u) = v^{B,i}_{R,A}$ and $R \in \texttt{confl}(R)$. Similar to the previous case.

  (b) $R(x, y) \wedge B(y) \rightarrow A(x)$ and a term u s.t. $R(t, u), B(u) \in \mathcal{M}_u$. Straightforward application of IH.
  (c) $B_1(x) \wedge \cdots \wedge B_n(x) \rightarrow A(x)$ s.t. $B_1(t), \ldots, B_n(t) \in \mathcal{M}_u$. Straightforward application of IH.
  (d) $A(x) \wedge x \approx y \rightarrow A(y)$ and a term u s.t. $A(u), u \approx t \in \mathcal{M}_u$. Straightforward application of IH.

(ii) Let $a = R(t_1, t_2)$. Then, there must be a rule in $\pi(\mathcal{K})^{\approx,\top}$:

  (a) $B(x) \rightarrow R(x, f^B_{R,A}(x)) \wedge A(f^A_{R,A}(x))$ and a term u such that $B(u) \in \mathcal{M}_u$ and $t = f^B_{R,B}(u)$. Similar to case (i)a.
  (b) $S(x, y) \rightarrow R(x, y)$. Straightforward application of IH.
  (c) $Inv(R)(y, x) \rightarrow R(x, y)$. Straightforward application of IH.
  (d) $R(x, y) \wedge y \approx z \rightarrow R(x, z)$ and a term u such that $R(t_1, u), u \approx t_2 \in \mathcal{M}_u$. Straightforward application of IH.
  (e) $R(x, y) \wedge x \approx z \rightarrow R(z, y)$ and a term u such that $R(u, t_2), u \approx t_1 \in \mathcal{M}_u$. Straightforward application of IH.

(iii) Let $a = t_1 \approx t_2$. Similar to case (ii). □

Lemma A.9. *Let σ and θ be as defined in Definition A.7. Then, for every $t_1, t_2 \in terms(\mathcal{M}_u)$ the following hold*

(i) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_1) < \sigma(t_2)$ *and* $\sigma(t_1) \notin N_I$ *implies* $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$
(ii) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_1) < \sigma(t_2)$ *and* $\sigma(t_1) \in N_I$ *implies either* $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$ *or* $R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$
(iii) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_1) \not< \sigma(t_2)$, $\sigma(t_1) \in N_I$ *and* $\sigma(t_2) \notin N_I$ *implies* $R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$
(iv) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_2) \not< \sigma(t_1)$, *and* $\sigma(t_2) \notin N_I$ *implies* $R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$
(v) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_2) < \sigma(t_1)$ *and* $\sigma(t_2) \in N_I$ *implies either* $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$ *or* $R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$
(vi) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_2) \not< \sigma(t_1)$, $\sigma(t_2) \in N_I$ *and* $\sigma(t_1) \notin N_I$ *implies* $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$

Proof. We show that the claims of the lemma hold by induction on the derivation level of atoms in $\mathcal{M}_u$. Let a be an atom in $\mathcal{M}_u$. We distinguish between these cases:

(i) $a = R(t_1, t_2) \in \mathcal{M}_u$ with $\sigma(t_1) < \sigma(t_2)$, and $\sigma(t_1) \notin N_I$. Then there must be a rule in $\pi(\mathcal{K})^{\approx,\top}$:

(a) $A(x) \rightarrow R(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$ with $A(t_1) \in \mathcal{M}_u$ and $t_2 = f^A_{R,B}(t_1)$. From Lemma A.8, $A(\theta(\sigma(t_1))) \in \mathcal{M}_c$ and one of the following holds:

– R is unsafe and $E_\mathcal{K}$ contains a rule of the form

$$A(x) \rightarrow R^f\left(x, f^A_{R,B}(x)\right) \wedge B\left(f^A_{R,B}(x)\right) \tag{45}$$

Then, $R^f(\theta(\sigma(t_1)), f^A_{R,B}(\theta(\sigma(t_1)))) \in \mathcal{M}_c$. By definition of θ, we have that $f^A_{R,B}(\theta(\sigma(t_1))) = \theta(f^A_{R,B}(\sigma(t_1)))$, and, from Lemma A.7, we can derive that $f^A_{R,B}(\sigma(t_1)) \approx \sigma(f^A_{R,B}(t_1)) \in \mathcal{M}_c$ and $f^A_{R,B}(t_1) = t_2$. Thus, $f^A_{R,B}(\theta(\sigma(t_1))) \approx \theta(\sigma(t_2))$ and $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$.

– R is safe and $E_\mathcal{K}$ contains a rule of the form

$$A(x) \wedge \texttt{notIn}\big(x, \texttt{unfold}(A, R, B)\big) \rightarrow R^f\left(x, v^{A,0}_{R,B}\right) \wedge B\left(v^{A,0}_{R,B}\right) \tag{46}$$

and $\theta(\sigma(t_1)) \notin \texttt{unfold}(A, R, B)$. Then, $R^f(\theta(\sigma(t_1)), v^{A,0}_{R,B}) \in \mathcal{M}_c$ and, by Lemma A.7, $\theta(\sigma(t_2)) = \theta(\sigma(f^A_{R,B}(t_1))) \approx \theta(f^A_{R,B}(\sigma(t_1))) = v^{A,0}_{R,B}$. Hence, $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$.

– R is safe and $E_\mathcal{K}$ contains a rule $A(x) \rightarrow R^f(x, v^{A,1}_{R,B}) \wedge B(v^{A,1}_{R,B})$ and $\theta(\sigma(t_1)) \in \texttt{cycle}(A, R, B)$. Similar to previous cases.

– $R \in \texttt{confl}(R)$ and $E_\mathcal{K}$ contains a rule $A(v^{A,i}_{R_B}) \rightarrow R^f(v^{A,i}_{R,B}, v^{A,i+1}_{R,B}) \wedge B(v^{A,i+1}_{R,B})$ and $\theta(\sigma(t_1)) = v^{A,i}_{R,B}$. Similar to previous cases.

(b) $S(x, y) \rightarrow R(x, y)$ with $S(t_1, t_2) \in \mathcal{M}_u$. By IH we can derive that $S^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$ and $E_\mathcal{K}$ contains a rule $S^f(x, y) \rightarrow R^f(x, y)$, thus $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$.

(c) $Inv(R)(y, x) \rightarrow R(x, y)$ with $Inv(R)(t_2, t_1) \in \mathcal{M}_u$. By IH we can derive $Inv(R)^b(\theta(\sigma(t_2)), \theta(\sigma(t_1))) \in \mathcal{M}_c$ and $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c$.

(d) $R(x, y), z \approx y \rightarrow R(x, z)$ and term t_3 s.t. $R(t_1, t_3), t_3 \approx t_2 \in \mathcal{M}_u$. Then, by Lemma A.7, $\sigma(t_3) \approx \sigma(t_2) \in \mathcal{M}_u$ and, by Lemma A.8, $\theta(\sigma(t_3)) \approx \theta(\sigma(t_2)) \in \mathcal{M}_c$. By IH over $R(t_1, t_3)$, we can deduce that $R^f(\theta(\sigma(t_1)), \theta(\sigma(t_3))) \in \mathcal{M}_c$.

(e) $R(x, y), z \approx x \rightarrow R(z, y)$ and term t_3 s.t. $R(t_3, t_2), t_3 \approx t_1 \in \mathcal{M}_u$. Similar to previous cases.

(ii) $a = R(t_1, t_2) \in \mathcal{M}_u$, with $\sigma(t_1) < \sigma(t_2)$, and $\sigma(t_1) \in N_I$ – similar to case (i).

(iii) $a = R(t_1, t_2) \in \mathcal{M}_u$, with $\sigma(t_1) \not< \sigma(t_2)$, $\sigma(t_1) \in N_I$ and $\sigma(t_2) \notin N_I$. Then, there must be a rule in $\pi(\mathcal{K})^{\approx,\top}$:

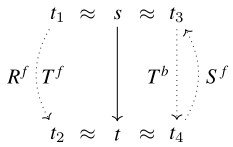


Fig. 14. Ambiguous roles in $\mathcal{M}_c$ in which both $T^f(s,t)$ and $T_b(s,t)$ hold.

(a) $A(x) \rightarrow R(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$, with $A(t_1) \in \mathcal{M}_u$ and $t_2 = f^A_{R,B}(t_1)$. But then, from Lemma A.7, it follows that either $\sigma(t_2) = h(\sigma(t_1))$, which contradicts the constraints on the derivation level of $\sigma(t_1), \sigma(t_2)$, or $\sigma(t_2) \in N_I$, which contradicts the assumption that $\sigma(t_2) \notin N_I$.
(b) $S(x, y) \rightarrow R(x, y)$ – similar to case (i)b.
(c) $Inv(R)(y, x) \rightarrow R(x, y)$ – similar to case (i)c.
(d) $R(x, y) \wedge y \approx z \rightarrow R(x, z)$ – similar to case (i)d.
(e) $R(x, y) \wedge x \approx z \rightarrow R(z, y)$ – similar to case (i)e.

(iv) $a = R(t_1, t_2) \in \mathcal{M}_u$, with $\sigma(t_2) \prec \sigma(t_1)$, and $\sigma(t_2) \notin N_I$ – similar to case (iii).
(v) $a = R(t_1, t_2) \in \mathcal{M}_u$, with $\sigma(t_2) \prec \sigma(t_1)$, and $\sigma(t_2) \in N_I$ – similar to case (ii).
(vi) $a = R(t_1, t_2) \in \mathcal{M}_u$, with $\sigma(t_2) \not\prec \sigma(t_1)$, $\sigma(t_2) \in N_I$ and $\sigma(t_1) \notin N_I$ – similar to case (i). □

Lemma A.10. *For every $t_1, t_2 \in terms(\mathcal{M}_c)$, $t_1 \approx t_2$ implies $depth_c(t_1) = depth_c(t_2)$.*

Proof. Trivially proven by observing that, if $t_1 \approx t_2$ then $\eta_c(t_1) = \eta_c(t_2)$ and, since Definition A.8 is based on η_c, $depth_c(t_1) = depth_c(t_2)$. □

Lemma A.11. *For every $t \in terms(\mathcal{M}_c)$, concepts A, B and role R, such that $v^{A,0}_{R,B} \not\approx a$, for every $a \in N_I$, it holds that:*

1. *$t \in \texttt{cycle}(A, R, B)$ and $R^f(t, v^{A,i}_{R,B}) \in \mathcal{M}_c$ implies $i = 1$;*
2. *$t \notin \texttt{cycle}(A, R, B)$ and $R^f(t, v^{A,i}_{R,B}) \in \mathcal{M}_c$ implies $i = 0$;*

Proof. By definition of canonical model and Definition 2.5. □

Lemma A.12. *For any role T and terms s and t, it is* not *the case that both $T^f(s,t) \in \mathcal{M}_c$ and $T^b(s,t) \in \mathcal{M}_c$.*

Proof. Assume the opposite. Then, there must be some roles R and S and terms t_1, t_2, t_3 and t_4, such that $R \sqsubseteq^*_{\mathcal{R}} T$, $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$, $t_1 \approx s, t_2 \approx t, t_3 \approx s, t_4 \approx t \in \mathcal{M}_c$, $R^f(t_1, t_2), S^f(t_4, t_3) \in \mathcal{M}_c$, t_2 is of type (A, R, B) and t_3 is of type (D, S, C) for some concept A, B, C and D (see Fig. 14).

We first deal with the case where one of t_1, t_2, t_3 and t_4 is equal to a named individual. w.l.o.g., let's assume that $t_1 \approx a \in \mathcal{M}_c$, with $a \in N_I$. Then, $t_3 \approx a \in \mathcal{M}_c$ as well. From the fact that $R(a, t_2) \in \mathcal{M}_c$ and Lemma A.1 it follows that $R(a, u^A_{R,B}) \in \mathcal{M}_{RSA}$. Furthermore, $S(t_4, t_3) \in \mathcal{M}_c$ implies $S(t_2, a) \in \mathcal{M}_c$, and thus $S(u^A_{R,B}, a) \in \mathcal{M}_{RSA}$. Since it holds that $R \sqsubseteq^*_{\mathcal{R}} T$ and $S \sqsubseteq^*_{\mathcal{R}} Inv(T)$, it follows that $\mathcal{K}$ is not *equality-safe* – contradiction.

In the following, we assume that none of t_1, t_2, t_3 or t_4 are equal to a named individual. Then one of the following holds:

– if t_1 is of the form $v^{D,i}_{S,C}$, then, from Lemma A.3, $t_3 = t_1$. We then distinguish between the following cases:

  * t_2 is of the form $v^{A,i}_{R,B}$. Then, by Lemma A.3, $t_4 = t_2$.
    If $(A, R, B) \prec (D, S, C)$ we have that either

$$\begin{cases} t_1 = v^{D,0}_{S,C} \\ t_2 = v^{A,1}_{R,B} \end{cases} \quad \text{or} \quad \begin{cases} t_1 = v^{D,1}_{S,C} \\ t_2 = v^{A,0}_{R,B} \end{cases} \tag{47}$$

and

$$\begin{cases} t_3 = v_{S,C}^{D,0} \\ t_4 = v_{R,B}^{A,0} \end{cases} \quad \text{or} \quad \begin{cases} t_3 = v_{S,C}^{D,1} \\ t_4 = v_{R,B}^{A,1} \end{cases} \tag{48}$$

But this is a contradiction to the fact that $t_1 = t_3$ and $t_2 = t_4$.
A similar derivation can be done if $(D, S, C) \prec (A, R, B)$.

* t_2 is of the form $f_{R,B}^A(t_1)$ and R is unsafe. Then, $S^f(f_{R,B}^A(t_1), t_1) = S^f(f_{R,B}^A(v_{S,C}^{D,i}), v_{S,C}^{D,i}) \in \mathcal{M}_c$. If $i = 0$, $f_{R,B}^A(v_{S,C}^{D_0}) \in \texttt{cycle}(D, S, C)$ and, from Lemma A.11, $S^f(f_{R,B}^A(v_{S,C}^{D,0}), v_{S,C}^{D,0}) \notin \mathcal{M}_c$ – contradiction. If $i = 1$, $f_{R,B}^A(v_{S,C}^{D,1}) \notin \texttt{cycle}(D, S, C)$. Thus, by Lemma A.11, we have that $S^f(f_{R,B}^A(v_{S,C}^{D,1}), v_{S,C}^{D,1}) \notin \mathcal{M}_c$ – contradiction.

– if both t_1 and t_2 are functional, t_3 and t_4 are functional as well and $t_2 = f_{R,B}^A(t_1)$ and $t_3 = f_{S,C}^D(t_4)$. From Lemma A.10, it follows that $depth_c(t_1) = depth_c(t_3)$ and $depth_c(t_2) = depth_c(t_4)$. But $depth_c(t_2) = depth_c(t_1) + 1$ and $depth_c(t_3) = depth_c(t_4) + 1$ – contradiction. □

Lemma A.13. *Let ρ be a non-anonymous match for q over $\mathcal{M}_u$ and let $\lambda(\cdot) = \theta(\sigma(\rho(\cdot)))$. Furthermore, let $\vec{a}$ be the following adornment for $\psi(\vec{x}, \vec{y})$:*

$$a_j = \begin{cases} \sqcup & \text{if } R_j(\lambda(u_{1j}), \lambda(u_{2j})) \in \mathcal{M}_c \text{ and } R_j^f(\lambda(u_{1j}), \lambda(u_{2j})), R_j^b(\lambda(u_{1j}), \lambda(u_{2j})) \notin \mathcal{M}_c \\ f & \text{if } R_j^f(\lambda(u_{1j}), \lambda(u_{2j})) \in \mathcal{M}_c \\ b & \text{if } R_j^b(\lambda(u_{1j}), \lambda(u_{2j})) \in \mathcal{M}_c \end{cases} \tag{49}$$

Then $(\lambda, \vec{a})$ is an adorned match for q over $\mathcal{M}_c$. Moreover, $(\lambda, \vec{a})$ is non-anonymous, fork-free and acyclic.

Proof. Following from Lemma A.9, $(\lambda, \vec{a})$ is an adorned match for q over $\mathcal{M}_c$. It is also easy to see that $(\lambda, \vec{a})$ is non-anonymous provided that ρ is non-anonymous.

To see that $(\lambda, \vec{a})$ is acyclic, assume the contrary. Then there exists a sequence

$$R_{o_1}^f(y_{l_1}, y_{l_2}), R_{o_2}^f(y_{l_3}, y_{l_4}), \ldots, R_{o_p}^f(y_{l_{2p-1}}, y_{l_{2p}}) \in \left(\psi(\vec{x}, \vec{y})\right)_n^{\vec{a}} \tag{50}$$

such that:

1. $id(\lambda(\vec{x}), \lambda(\vec{y}), l_{2i}, l_{2i+1}) \in \mathcal{M}$ for every $1 \leqslant i \leqslant p$ where $l_{2p+1} = l_1$;
2. $\texttt{NI}(\lambda(y_{l_j})) \notin \mathcal{M}$ for every $1 \leqslant j \leqslant 2p$.

Let $s_i = \sigma(\rho(y_{l_{2i}}))$ for $1 \leqslant i \leqslant p$. Then

$$R_{o_1}(s_p, s_1), R_{o_2}(s_1, s_2), \ldots, R_{o_p}(s_{p-1}, s_p) \in \mathcal{M}_u \tag{51}$$

where $s_i \notin N_I$ for every $1 \leqslant i \leqslant p$. Then, by Lemma A.8, Lemma A.9 and Lemma A.12 and from the fact that $R_{o_i}^f(\theta(s_i), \theta(s_{i+1})) \in \mathcal{M}_c$ for every $1 \leqslant i \leqslant p$, it follows that $s_i \prec s_{i+1}$, for every $1 \leqslant i \leqslant p$. But then $s_i \prec s_i$ holds, which is a contradiction.

To see that $(\lambda, \vec{a})$ is fork-free, we assume again the contrary. Then, there must exist axioms $R^f(u, y_i), S^f(v, y_j) \in (\psi(\vec{x}, \vec{y}))_n^{\vec{a}}$, such that $u, v \in \vec{x} \cup \vec{y}$, $y_i, y_j \in \vec{y}$ and $id(\lambda(\vec{x}), \lambda(\vec{y}), i, j) \in \mathcal{M}$ and $\lambda(u) \not\approx \lambda(v)$.

From the fact that $id(\lambda(\vec{x}), \lambda(\vec{y}), i, j) \in \mathcal{M}$, it follows that either:

– $i = j$: in this scenario, since $\texttt{NI}(\lambda(y_i)), \texttt{NI}(\lambda(y_j)) \notin \mathcal{M}$, it follows that $\texttt{NI}(\sigma(\rho(y_i))), \texttt{NI}(\sigma(\rho(y_j))) \notin \mathcal{M}_u$, $\sigma(\rho(y_i)) = f_{R,B}(\sigma(\rho(u)))$ and $\sigma(\rho(y_j)) = f_{S,C}(\sigma(\rho(v)))$. But, as $i = j$, we have $\sigma(\rho(y_i)) = \sigma(\rho(y_j))$, $f_{R,B}$, $f_{S,C}$ are the same function symbol and $\sigma(\rho(u)) = \sigma(\rho(v))$. Then $\lambda(u) = \lambda(v)$ – contradiction.
– or there exist two sequences of atoms:

* $R^f_{l_1}(y_i, y_{l_1}), \ldots, R^f_{l_m}(y_{l_{m-1}}, y_{l_m})$
* $R^f_{k_1}(y_j, y_{k_1}), \ldots, R^f_{k_m}(y_{k_{m-1}}, y_{k_m})$

such that $l_m = k_m$ and $id(\lambda(\vec{x}), \lambda(\vec{y}), l_i, k_i) \in \mathcal{M}$, for every $1 \leqslant i \leqslant m$.
Then, it can be shown by induction on the length m of the sequences introduced above that $\sigma(\rho(y_{l_i})) = \sigma(\rho(y_{k_i}))$, for every $1 \leqslant i \leqslant m$ and that $\sigma(\rho(u)) = \sigma(\rho(v))$. Finally, we obtain $\lambda(u) = \lambda(v)$ – contradiction. □

Theorem 5.1. *Let $\mathcal{K}$ be a satisfiable $\mathcal{ALCHOIQ}^+$ KB and $\mathcal{K}' = \mathtt{upper}(\delta(\mathcal{K}))$. Moreover, let $q(\vec{x}) = \exists \vec{y} \varphi(\vec{x}, \vec{y})$ be a CQ. Then,*

(i) *$\mathcal{K}'$ is* RSA$^+$,
(ii) *$cert(q, \mathcal{K}) \subseteq cert(q, \mathcal{K}')$,*
(iii) *if $\vec{x} \in cert(q, \mathcal{K})$ then $\mathcal{P}_{\mathcal{K}',q} \models \mathtt{Ans}(\vec{x})$.*

Proof. Consider the following

(i) Both the construction of $G_{\mathcal{K}}$ and the definition of equality safety are expressed in a purely syntactical way. It is easy to see that rewriting the axioms (T4) and (T5), as defined in Def. 5.2, is enough to render the knowledge base RSA$^+$.
(ii) In order to prove that $cert(q, \mathcal{K}) \subseteq cert(q, \mathcal{K}')$, we will show that for every model $\mathcal{I}$ such that $\mathcal{I}$ is a model of $\mathcal{K}'$, $\mathcal{I}$ is a model of $\mathcal{K}$.[29]
Given a model $\mathcal{I}$ for $\mathcal{K}'$, we know that there are four possible ways in which $\mathcal{K}'$ differs from $\mathcal{K}$:
   (a) An axiom $\alpha \equiv A \sqsubseteq \exists R.B \in \mathcal{K}$ has been rewritten into $\beta \equiv A \sqsubseteq \exists R.\{b^A_{R,B}\}$ and $B(b^A_{R,B})$. Since we have that $\mathcal{I} \models \beta$, we know that for every $a \in A^{\mathcal{I}}$, we have $(a, b^A_{R,B}) \in R^{\mathcal{I}}$ and $b^A_{R,B} \in B^{\mathcal{I}}$. But then $\mathcal{I}$ is also a model of the KB where β has been substituted with α.
   (b) An axiom $\alpha \equiv C \sqsubseteq \leqslant 1S.D \in \mathcal{K}$ has been rewritten into $\beta \equiv C \sqcap \exists S.D \sqsubseteq \bot_f$. Since we have that $\mathcal{I} \models \beta$, we know that for every $c \in C^{\mathcal{I}}$, there is no individual d such that $(c, d) \in S^{\mathcal{I}}$ and $d \in D^{\mathcal{I}}$. But then $\mathcal{I}$ is also a model of the KB where β has been substituted with α.
   (c) An axiom $\alpha \equiv A \sqsubseteq \leqslant mR.B \in \mathcal{K}$ has been rewritten into $\beta \equiv A \sqsubseteq \leqslant 1R.B$. Similar to the previous steps.
   (d) An axiom $\alpha \equiv \bigsqcap_{i=1}^n A_i \sqsubseteq \bigsqcup_{j=1}^m B_j$ has been rewritten into $\beta \equiv \bigsqcap_{i=1}^n A_i \sqsubseteq B$ with $B = \mathtt{ch}(\{B_j \mid 1 \leqslant j \leqslant m\})$. Similar to the previous steps.

(iii) Assume $\vec{x} \in cert(q, \mathcal{K})$. By step (ii) we know that $\vec{x} \in cert(q, \mathcal{K}')$. Then according to Lemma A.6, there exists a match ρ for q over $\mathcal{M}_u$. According to Lemma A.13 one can construct from ρ a match $(\lambda, \vec{a})$ over $\mathcal{M}_c$ which is non-anonymous, fork-free and acyclic. Note that λ does not necessarily preserve the mapping of ρ over $terms(q) \setminus \vec{y}$, since λ is based on the definition of σ, which maps over representatives of a certain equivalence class induced by $\approx$. λ can be transformed into another mapping λ' such that

$$\lambda'(t) = \begin{cases} \rho(t) & \text{for every } t \in terms(q) \setminus \vec{y} \\ t & otherwise \end{cases} \tag{52}$$

It can be checked that $(\lambda', \vec{a})$ is still non-anonymous, fork-free and acyclic (intuitively because we are only updating the non-anonymous part). Then, by applying Lemma A.5, we obtain that $\mathcal{P}_{\mathcal{K}',q} \models \mathtt{Ans}(\lambda(\vec{x}))$. □

[29] Here we are using an alternative, but equivalent, definition of certain answer. Given a query $q(\vec{x})$ and a KB $\mathcal{K}$, $\vec{a} \in \mathtt{cert}(q, \mathcal{K})$ iff $\mathcal{K}, \mathcal{I} \models q(\vec{a})$ for every model $\mathcal{I}$ of $\mathcal{K}$.

Acknowledgements

This work was supported by the AIDA project (Alan Turing Institute, EP/N510129/1), the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project number 237889), Samsung Research UK, Siemens AG, and the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1) and UK FIRES (EP/S019111/1). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

[1] S. Abiteboul, R. Hull and V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995, http://webdam.inria.fr/Alice/. ISBN 0-201-53771-0.

[2] A. Acciarri, D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, M. Palmieri and R. Rosati, QuOnto: Querying ontologies, in: *Proceedings, the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, Pittsburgh, Pennsylvania, USA, July 9–13, 2005, M.M. Veloso and S. Kambhampati, eds, AAAI Press / The MIT Press, 2005, pp. 1670–1671, http://www.aaai.org/Library/AAAI/2005/isd05-001.php.

[3] C. Alrabbaa, S. Borgwardt, P. Koopmann and A. Kovtunova, Explaining ontology-mediated query answers using proofs over universal models, in: *Rules and Reasoning – 6th International Joint Conference on Rules and Reasoning, RuleML+RR 2022, Proceedings*, Berlin, Germany, September 26–28, 2022, G. Governatori and A. Turhan, eds, Lecture Notes in Computer Science, Vol. 13752, Springer, 2022, pp. 167–182. doi:10.1007/978-3-031-21541-4_11.

[4] F. Baader, S. Brandt and C. Lutz, Pushing the EL envelope, in: *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Scotland, UK, July 30–August 5, 2005, L.P. Kaelbling and A. Saffiotti, eds, Professional Book Center, 2005, pp. 364–369, http://ijcai.org/Proceedings/05/Papers/0372.pdf.

[5] F. Baader, I. Horrocks, C. Lutz and U. Sattler, *An Introduction to Description Logic*, Cambridge University Press, 2017.

[6] F. Baader, C. Lutz and S. Brandt, Pushing the EL envelope further, in: *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, Washington, DC, USA, 1–2 April 2008, K. Clark and P.F. Patel-Schneider, eds, CEUR Workshop Proceedings, Vols 496, CEUR-WS.org, 2008, http://ceur-ws.org/Vol-496/owled2008dc_paper_3.pdf.

[7] B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev and R. Velkov, OWLIM: A family of scalable semantic repositories, *Semantic Web Journal* **2**(1) (2011), 33–42. doi:10.3233/SW-2011-0026.

[8] J. Broekstra, A. Kampman and F. van Harmelen, Sesame: A generic architecture for storing and querying RDF and RDF schema, in: *The Semantic Web – ISWC 2002, First International Semantic Web Conference, Proceedings*, Sardinia, Italy, June 9–12, 2002, I. Horrocks and J.A. Hendler, eds, Lecture Notes in Computer Science, Vol. 2342, Springer, 2002, pp. 54–68. doi:10.1007/3-540-48005-6_7.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini and R. Rosati, Data complexity of query answering in description logics, in: *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning*, Lake District of the United Kingdom, June 2–5, 2006, AAAI Press, 2006, pp. 260–270.

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini and R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-lite family, *Journal of Automated Reasoning* **39**(3) (2007), 385–429. doi:10.1007/s10817-007-9078-x.

[11] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi and D.F. Savo, The MASTRO system for ontology-based data access, *Semantic Web Journal* **2**(1) (2011), 43–53. doi:10.3233/SW-2011-0029.

[12] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini and R. Rosati, Data complexity of query answering in description logics, *Artificial Intelligence* **195** (2013), 335–360. doi:10.1016/j.artint.2012.10.003.

[13] D. Carral, I. Dragoste and M. Krötzsch, The combined approach to query answering in horn-ALCHOIQ, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018*, Tempe, Arizona, 30 October–2 November 2018, M. Thielscher, F. Toni and F. Wolter, eds, AAAI Press, 2018, pp. 339–348, https://aaai.org/ocs/index.php/KR/KR18/paper/view/18076.

[14] D. Carral, C. Feier, B.C. Grau, P. Hitzler and I. Horrocks, Pushing the boundaries of tractable ontology reasoning, in: *Proceedings, Part II*, The Semantic Web – ISWC 2014 – 13th International Semantic Web Conference, Riva del Garda, Italy, October 19–23, 2014, Lecture Notes in Computer Science, Vol. 8797, Springer, 2014, pp. 148–163.

[15] S. Das, S. Sundara and R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, W3C, 2012, https://www.w3.org/TR/2012/REC-r2rml-20120927/.

[16] J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg, K. Srinivas and L. Ma, Scalable semantic retrieval through summarization and refinement, in: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada, July 22–26, 2007, AAAI Press, 2007, pp. 299–304, http://www.aaai.org/Library/AAAI/2007/aaai07-046.php.

[17] J. Dolby, A. Fokoue, A. Kalyanpur, E. Schonberg and K. Srinivas, Scalable highly expressive reasoner (SHER), *Journal of Web Semantics* **7**(4) (2009), 357–361. doi:10.1016/j.websem.2009.05.002.

[18] T. Eiter, M. Fink, H. Tompits and S. Woltran, On eliminating disjunctions in stable logic programming, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, Whistler, Canada, June 2–5, 2004, D. Dubois, C.A. Welty and M. Williams, eds, AAAI Press, 2004, pp. 447–458, http://www.aaai.org/Library/KR/2004/kr04-047.php.

[19] T. Eiter, M. Ortiz, M. Simkus, T. Tran and G. Xiao, Towards practical query answering for horn-SHIQ, in: *Proceedings of the 2012 International Workshop on Description Logics, DL-2012*, Rome, Italy, June 7–10, 2012, Y. Kazakov, D. Lembo and F. Wolter, eds, CEUR Workshop Proceedings, Vols 846, CEUR-WS.org, 2012, http://ceur-ws.org/Vol-846/paper_20.pdf.

[20] T. Eiter, M. Ortiz, M. Simkus, T. Tran and G. Xiao, Query rewriting for horn-SHIQ plus rules, in: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto, Ontario, Canada, July 22–26, 2012, J. Hoffmann and B. Selman, eds, AAAI Press, 2012, http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4931.

[21] O. Erling and I. Mikhailov, Virtuoso: RDF support in a native RDBMS, in: *Semantic Web Information Management – A Model-Based Perspective*, R.D. Virgilio, F. Giunchiglia and L. Tanca, eds, Springer, 2009, pp. 501–519. doi:10.1007/978-3-642-04329-1_21.

[22] C. Feier, D. Carral, G. Stefanoni, B.C. Grau and I. Horrocks, The combined approach to query answering beyond the OWL 2 profiles, in: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires*, Argentina, July 25–31, 2015, AAAI Press, 2015, pp. 2971–2977.

[23] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari and L. Schneider, Sweetening ontologies with DOLCE, in: *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Proceedings*, Siguenza, Spain, October 1–4, 2002, A. Gómez-Pérez and V.R. Benjamins, eds, Lecture Notes in Computer Science, Vol. 2473, Springer, 2002, pp. 166–181. doi:10.1007/3-540-45810-7_18.

[24] B. Glimm, I. Horrocks, B. Motik, G. Stoilos and Z. Wang, HermiT: An OWL 2 reasoner, *Journal of Automated Reasoning* **53**(3) (2014), 245–269. doi:10.1007/s10817-014-9305-1.

[25] B. Glimm, I. Horrocks and U. Sattler, Conjunctive query entailment for SHOQ, in: *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, Brixen-Bressanone, Near Bozen-Bolzano, Italy, 8–10 June, 2007, D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A. Turhan and S. Tessaris, eds, CEUR Workshop Proceedings, Vols 250, CEUR-WS.org, 2007, http://ceur-ws.org/Vol-250/paper_63.pdf.

[26] B. Glimm, Y. Kazakov, I. Kollia and G.B. Stamou, OWL query answering based on query extension, in: *Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014) Co-Located with 13th International Semantic Web Conference on (ISWC 2014)*, Riva del Garda, Italy, October 17–18, 2014, C.M. Keet and V.A.M. Tamma, eds, CEUR Workshop Proceedings, Vol. 1265, CEUR-WS.org, 2014, pp. 1–12, http://ceur-ws.org/Vol-1265/owled2014_submission_1.pdf.

[27] B. Glimm, Y. Kazakov, I. Kollia and G.B. Stamou, Lower and upper bounds for SPARQL queries over OWL ontologies, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, January 25–30, 2015, B. Bonet and S. Koenig, eds, AAAI Press, 2015, pp. 109–115, http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9715.

[28] B. Glimm, C. Lutz, I. Horrocks and U. Sattler, Conjunctive query answering for the description logic SHIQ, *Journal of Artificial Intelligence Research* **31** (2008), 157–204. doi:10.1613/jair.2372.

[29] B.N. Grosof, I. Horrocks, R. Volz and S. Decker, Description logic programs: Combining logic programs with description logic, in: *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003*, Budapest, Hungary, May 20–24, 2003, ACM, 2003, pp. 48–57.

[30] Y. Guo, Z. Pan and J. Heflin, LUBM: A benchmark for OWL knowledge base systems, *Journal of Web Semantics* **3**(2–3) (2005), 158–182. doi:10.1016/j.websem.2005.06.005.

[31] V. Haarslev, K. Hidde, R. Möller and M. Wessel, The RacerPro knowledge representation and reasoning system, *Semantic Web* **3**(3) (2012), 267–277. doi:10.3233/SW-2011-0032.

[32] A. Haga, C. Lutz, L. Sabellek and F. Wolter, How to approximate ontology-mediated queries, in: *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, Online Event, November 3–12, 2021, 2021, pp. 323–333. doi:10.24963/kr.2021/31.

[33] S. Harris and A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation, W3C, 2013, https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[34] P. Hitzler, M. Krötzsch, S. Rudolph and T. Tserendorj, Approximate OWL instance retrieval with SCREECH, in: *Logic and Probability for Scene Interpretation, 24.02.–29.02.2008*, A.G. Cohn, D.C. Hogg, R. Möller and B. Neumann, eds, Dagstuhl Seminar Proceedings, Vol. 08091, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2008, http://drops.dagstuhl.de/opus/volltexte/2008/1615/.

[35] M. Horridge and S. Bechhofer, The OWL API: A Java API for OWL ontologies, *Semantic Web* **2**(1) (2011), 11–21. doi:10.3233/SW-2011-0025.

[36] I. Horrocks and S. Tessaris, A conjunctive query language for description logic aboxes, in: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, Austin, Texas, USA, July 30–August 3, 2000, H.A. Kautz and B.W. Porter, eds, AAAI Press / The MIT Press, 2000, pp. 399–404, http://www.aaai.org/Library/AAAI/2000/aaai00-061.php.

[37] D. Hovland, R. Kontchakov, M.G. Skjæveland, A. Waaler and M. Zakharyaschev, Ontology-based data access to slegge, in: *Proceedings, Part II*, The Semantic Web – ISWC 2017 – 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Lecture Notes in Computer Science, Vol. 10588, Springer, 2017, pp. 120–129.

[38] P. Hu, B. Motik and I. Horrocks, Modular materialisation of datalog programs, in: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, the Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, the Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, Honolulu, Hawaii, USA, January 27–February 1, 2019, AAAI Press, 2019, pp. 2859–2866. doi:10.1609/aaai.v33i01.33012859.

[39] F. Igne, S. Germano and I. Horrocks, RSAComb: Combined approach for CQ answering in RSA, in: *Proceedings of the 34th International Workshop on Description Logics (DL 2021) Part of Bratislava Knowledge September (BAKS 2021)*, Bratislava, Slovakia, September 19th to 22nd, 2021, M. Homola, V. Ryzhikov and R.A. Schmidt, eds, CEUR Workshop Proceedings, Vols 2954, CEUR-WS.org, 2021, http://ceur-ws.org/Vol-2954/paper-18.pdf.

[40] F. Igne, S. Germano and I. Horrocks, Computing CQ lower-bounds over OWL 2 through approximation to RSA, in: *The Semantic Web – ISWC 2021 – 20th International Semantic Web Conference, ISWC 2021, Proceedings*, Virtual Event, October 24–28, 2021, A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P.M. Barnaghi, A. Haller, M. Dragoni and H. Alani, eds, Lecture Notes in Computer Science, Vol. 12922, Springer, 2021, pp. 200–216. doi:10.1007/978-3-030-88361-4_12.

[41] F. Igne, S. Germano and I. Horrocks, RSAComb – Combined approach for Conjunctive Query answering in RSA, Zenodo, 2022. doi:10.5281/zenodo.6564261.

[42] F. Igne, S. Germano and I. Horrocks, ACQuA – A hybrid framework providing a CQ answering service for OWL, Zenodo, 2022. doi:10.5281/zenodo.6564388.

[43] F. Igne, S. Germano and I. Horrocks, Benchmarks and scripts for ACQuA and RSAComb, *Zenodo* (2022). doi:10.5281/zenodo.6564995.

[44] E. Kharlamov, D. Hovland, E. Jiménez-Ruiz, D. Lanti, H. Lie, C. Pinkel, M. Rezk, M.G. Skjæveland, E. Thorstensen, G. Xiao, D. Zheleznyakov and I. Horrocks, Ontology based access to exploration data at statoil, in: *11–15, 2015, Proceedings, Part II*, The Semantic Web – ISWC 2015 – 14th International Semantic Web Conference, Lecture Notes in Computer Science, Vol. 9367, Springer, Bethlehem, PA, USA, 2015, pp. 93–112.

[45] I. Kollia and B. Glimm, Optimizing SPARQL query answering over OWL ontologies, *Journal of Artificial Intelligence Research* **48** (2013), 253–303. doi:10.1613/jair.3872.

[46] R. Kontchakov, C. Lutz, D. Toman, F. Wolter and M. Zakharyaschev, 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, May 9–13, 2010, AAAI Press, 2010.

[47] R. Kontchakov, C. Lutz, D. Toman, F. Wolter and M. Zakharyaschev, The combined approach to ontology-based data access, in: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, Barcelona, Catalonia, Spain, July 16–22, 2011, T. Walsh, ed., IJCAI/AAAI, 2011, pp. 2656–2661. doi:10.5591/978-1-57735-516-8/IJCAI11-442.

[48] M. Krötzsch, S. Rudolph and P. Hitzler, Conjunctive Queries for a Tractable Fragment of OWL 1.1, 2007, pp. 310–323. ISBN 978-3-540-76297-3.

[49] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri and F. Scarcello, The DLV system for knowledge representation and reasoning, *ACM Transactions on Computational Logic* **7**(3) (2006), 499–562. doi:10.1145/1149114.1149117.

[50] C. Lutz, I. Seylan, D. Toman and F. Wolter, The combined approach to OBDA: taming role hierarchies using filters, in: *The Semantic Web – ISWC 2013 – 12th International Semantic Web Conference, Proceedings, Part I*, Sydney, NSW, Australia, October 21–25, 2013, H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J.X. Parreira, L. Aroyo, N.F. Noy, C. Welty and K. Janowicz, eds, Lecture Notes in Computer Science, Vol. 8218, Springer, 2013, pp. 314–330. doi:10.1007/978-3-642-41335-3_20.

[51] C. Lutz, D. Toman and F. Wolter, Conjunctive query answering in the description logic EL using a relational database system, in: *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence*, Pasadena, California, USA, July 11–17, 2009, C. Boutilier, eds, 2009, pp. 2070–2075, http://ijcai.org/Proceedings/09/Papers/341.pdf.

[52] B. McBride, Jena: Implementing the RDF model and syntax specification, in: *Proceedings of the Second International Workshop on the Semantic Web – SemWeb'2001*, Hongkong, China, May 1, 2001, S. Decker, D.A. Fensel, A.P. Sheth and S. Staab, eds, CEUR Workshop Proceedings, Vols 40, CEUR-WS.org, 2001, http://CEUR-WS.org/Vol-40/mcbride.pdf.

[53] B. Motik, B.C. Grau, I. Horrocks, Z. Wu, A. Fokoue and C. Lutz, OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation, W3C, 2012, https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.

[54] B. Motik, Y. Nenov, R. Piro and I. Horrocks, Handling owl: SameAs via rewriting, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, January 25–30, 2015, B. Bonet and S. Koenig, eds, AAAI Press, 2015, pp. 231–237.

[55] B. Motik, Y. Nenov, R. Piro and I. Horrocks, Incremental update of datalog materialisation: The backward/forward algorithm, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, January 25–30, 2015, B. Bonet and S. Koenig, eds, AAAI Press, 2015, pp. 1560–1568.

[56] B. Motik, Y. Nenov, R. Piro and I. Horrocks, Combining rewriting and incremental materialisation maintenance for datalog programs with equality, in: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires*, Argentina, July 25–31, 2015, Q. Yang and M.J. Wooldridge, eds, AAAI Press, 2015, pp. 3127–3133.

[57] B. Motik, Y. Nenov, R. Piro, I. Horrocks and D. Olteanu, Parallel materialisation of datalog programs in centralised, main-memory RDF systems, in: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Québec City, Québec, Canada, July 27–31, 2014, C.E. Brodley and P. Stone, eds, AAAI Press, 2014, pp. 129–137.

[58] B. Motik, P. Patel-Schneider and B. Parsia, OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation, W3C, 2012, https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[59] B. Motik, R. Shearer and I. Horrocks, Hypertableau reasoning for description logics, *Journal of Artificial Intelligence Research* **36** (2009), 165–228. doi:10.1613/jair.2811.

[60] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu and J. Banerjee, RDFox: A highly-scalable RDF store, in: *11–15, 2015, Proceedings, Part II*, The Semantic Web – ISWC 2015 – 14th International Semantic Web Conference, M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, eds, Lecture Notes in Computer Science, Vol. 9367, Springer, Bethlehem, PA, USA, 2015, pp. 3–20.

[61] G. Orsi and A. Pieris, Optimizing query answering under ontological constraints, *Proceedings of the VLDB Endowment* **4**(11) (2011), 1004–1015, http://www.vldb.org/pvldb/vol4/p1004-orsi.pdf. doi:10.14778/3402707.3402737.

[62] M. Ortiz and M. Simkus, Reasoning and query answering in description logics, in: *Reasoning Web. Semantic Technologies for Advanced Query Answering – 8th International Summer School 2012, Proceedings*, Vienna, Austria, September 3–8, 2012, T. Eiter and T. Krennwallner, eds, Lecture Notes in Computer Science, Vol. 7487, Springer, 2012, pp. 1–53. doi:10.1007/978-3-642-33158-9_1.

[63] J.Z. Pan and E. Thomas, Approximating OWL-DL ontologies, in: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada, July 22–26, 2007, AAAI Press, 2007, pp. 1434–1439.

[64] H. Pérez-Urbina, I. Horrocks and B. Motik, Efficient query answering for OWL 2, in: *The Semantic Web – ISWC 2009, 8th International Semantic Web Conference, ISWC 2009*, Chantilly, VA, USA, October 25–29, 2009, Proceedings, Lecture Notes in Computer Science, Vol. 5823, Springer, 2009, pp. 489–504. doi:10.1007/978-3-642-04930-9_31.

[65] H. Pérez-Urbina, B. Motik and I. Horrocks, Tractable query answering and rewriting under description logic constraints, *J. Appl. Log.* **8**(2) (2010), 186–209. doi:10.1016/j.jal.2009.09.004.

[66] A. Poggi, D. Lembo, D. Calvanese, G.D. Giacomo, M. Lenzerini and R. Rosati, Linking data to ontologies, *J. Data Semant.* **10** (2008), 133–173. doi:10.1007/978-3-540-77688-8_5.

[67] Y. Ren, G. Gröner, J. Lemcke, T. Rahmani, A. Friesen, Y. Zhao, J.Z. Pan and S. Staab, Validating process refinement with ontologies, in: *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, Oxford, UK, July 27–30, 2009, B.C. Grau, I. Horrocks, B. Motik and U. Sattler, eds, CEUR Workshop Proceedings, Vol. 477, CEUR-WS.org, http://ceur-ws.org/Vol-477/paper_59.pdf.

[68] Y. Ren, J.Z. Pan, I. Guclu and M.J. Kollingbaum, A combined approach to incremental reasoning for EL ontologies, in: *Web Reasoning and Rule Systems – 10th International Conference, RR 2016*, Aberdeen, UK, September 9–11, 2016, Proceedings, Lecture Notes in Computer Science, Vol. 9898, Springer, 2016, pp. 167–183. doi:10.1007/978-3-319-45276-0_13.

[69] R. Rosati, On conjunctive query answering in EL, in: *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, Brixen-Bressanone, Near Bozen-Bolzano, Italy, 8–10 June, 2007, D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A. Turhan and S. Tessaris, eds, CEUR Workshop Proceedings, Vols 250, CEUR-WS.org, 2007, http://ceur-ws.org/Vol-250/paper_83.pdf.

[70] B. Selman and H.A. Kautz, Knowledge compilation and theory approximation, *Journal of the ACM* **43**(2) (1996), 193–224. doi:10.1145/226643.226644.

[71] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur and Y. Katz, Pellet: A practical OWL-DL reasoner, *Journal of Web Semantics* **5**(2) (2007), 51–53. doi:10.1016/j.websem.2007.03.004.

[72] G. Stefanoni and B. Motik, Answering conjunctive queries over EL knowledge bases with transitive and reflexive roles, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, January 25–30, 2015, B. Bonet and S. Koenig, eds, AAAI Press, 2015, pp. 1611–1617, http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9310.

[73] G. Stefanoni, B. Motik and I. Horrocks, Introducing nominals to the combined query answering approaches for EL, in: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, Bellevue, Washington, USA, July 14–18, 2013, M. desJardins and M.L. Littman, eds, AAAI Press, 2013, http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6156.

[74] A. Steigmiller and B. Glimm, Absorption-based query entailment checking for expressive description logics, in: *Proceedings of the 32nd International Workshop on Description Logics*, Oslo, Norway, June 18–21, 2019, M. Simkus and G.E. Weddell, eds, CEUR Workshop Proceedings, Vols 2373, CEUR-WS.org, 2019, http://ceur-ws.org/Vol-2373/paper-25.pdf.

[75] A. Steigmiller and B. Glimm, Parallelised ABox reasoning and query answering with expressive description logics, in: *The Semantic Web – 18th International Conference, ESWC 2021, Virtual Event*, Proceedings, June 6–10, 2021, R. Verborgh, K. Hose, H. Paulheim, P. Champin, M. Maleshkova, Ó. Corcho, P. Ristoski and M. Alam, eds, Lecture Notes in Computer Science, Vol. 12731, Springer, 2021, pp. 23–39. doi:10.1007/978-3-030-77385-4_2.

[76] A. Steigmiller, B. Glimm and T. Liebig, Reasoning with nominal schemas through absorption, *Journal of Automated Reasoning* **53**(4) (2014), 351–405. doi:10.1007/s10817-014-9310-4.

[77] A. Steigmiller, T. Liebig and B. Glimm, Konclude: System description, *Journal of Web Semantics (JWS)* **27** (2014), 78–85. doi:10.1016/j.websem.2014.06.003.

[78] G. Stoilos, Hydrowl: A hybrid query answering system for OWL 2 DL ontologies, in: *Web Reasoning and Rule Systems – 8th International Conference, RR 2014, Proceedings*, Athens, Greece, September 15–17, 2014, R. Kontchakov and M. Mugnier, eds, Lecture Notes in Computer Science, Vol. 8741, Springer, 2014, pp. 230–238. doi:10.1007/978-3-319-11113-1_20.

[79] G. Stoilos, Ontology-based data access using rewriting, OWL 2 RL systems and repairing, in: *The Semantic Web: Trends and Challenges – 11th International Conference, ESWC 2014, Proceedings*, Anissaras, Crete, Greece, May 25–29, 2014, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab and A. Tordai, eds, Lecture Notes in Computer Science, Vol. 8465, Springer, 2014, pp. 317–332. doi:10.1007/978-3-319-07443-6_22.

[80] E. Thomas, J.Z. Pan and Y. Ren, TrOWL: Tractable OWL 2 reasoning infrastructure, in: *Proceedings, Part II*, The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30–June 3, 2010, L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral and T. Tudorache, eds, Lecture Notes in Computer Science, Vol. 6089, Springer, 2010, pp. 431–435. doi:10.1007/978-3-642-13489-0_38.

[81] A.D. Val, First order LUB approximations: Characterization and algorithms, *Artif. Intell.* **162**(1–2) (2005), 7–48. doi:10.1016/j.artint.2004.01.003.

[82] S. Wandelt, R. Möller and M. Wessel, Towards scalable instance retrieval over ontologies, *International Journal of Software and Informatics* **4**(3) (2010), 201–218, http://www.ijsi.org/ch/reader/view_abstract.aspx?file_no=i59.

[83] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati and M. Zakharyaschev, Ontology-based data access: A survey, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, Stockholm, Sweden, July 13–19, 2018, ijcai.org, 2018, pp. 5511–5519.

[84] Y. Zhou, PAGOdA: Pay-as-you-go ontology query answering using a datalog reasoner, PhD thesis, University of Oxford, UK, 2015, https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.711817.

[85] Y. Zhou, B.C. Grau, Y. Nenov, M. Kaminski and I. Horrocks, PAGOdA: Pay-as-you-go ontology query answering using a datalog reasoner, *Journal of Artificial Intelligence Research* **54** (2015), 309–367. doi:10.1613/jair.4757.